# ST7MDT6-EMU2B
# HDS2 Series Emulator
# User Manual

## Release 1.2

## May 2000

Ref: DOC-ST7MDT6-EMU2B

# Table of Contents

# Table of Contents

# 1        INTRODUCTION

Thanks for choosing ST7! This manual will help you get started with the ST7MDT6-EMU2B.

The ST7MDT6-EMU2B emulator allows you to control the execution of programs that you have written for the **ST72C171** MCU and assists you in debugging your application hardware as well as your software. The ST7MDT6-EMU2B kit comes with a new debugger software package—ST7 Visual Debug—which contains all of the necessary resources to help you design, develop and debug ST7 application software running in a real environment.

*Note:*     *If you come across any terms or abbreviations you do not understand, you can check their meaning in the Glossary on page 57.*

The Emulator Package is made up of two main parts:

• The Hardware Development System (**ST7-HDS2**), which is the common mainframe to all ST7 emulators.

• The **ST7MDT6-Active Probe**, dedicated to the family, which constitutes the physical link between the emulator and your application.

*Note:*     *When receiving the ST7MDT6-EMU2B development tool, please refer to the Delivery Checklist on page 9 to confirm that all of the contents of the package are present.*

The emulator performs two main functions:

• It replaces the microcontroller in the application, by means of an emulation probe that is plugged into the application in place of the emulated MCU.

• It controls the internal data bus of the emulated microcontroller, providing arbitration and tracing capabilities on all accesses to any of the following resources:

    -   ST7-HDS2 resources,

    -   ST7MDT6-Active Probe resources,

    -   Application resources.

Therefore, you can have the emulator running your software in the application just as the emulated MCU would do, and have extensive tracing and control capabilities (i.e. keeping track what the MCU does, and making it react in a specific way upon defined conditions).

In this way, it is possible to fully emulate microcontroller resources.

## 1.1        Emulator Configuration

*Figure 1* shows the general configuration of the ST7MDT6-EMU2B emulator kit. The main ST7-HDS2 box is connected to your PC via the parallel port. Two flat cables connect the ST7-HDS2 box to the ST7MDT6-Active Probe, to which a device adapter can be fixed so that you can connect the emulator to your application board.


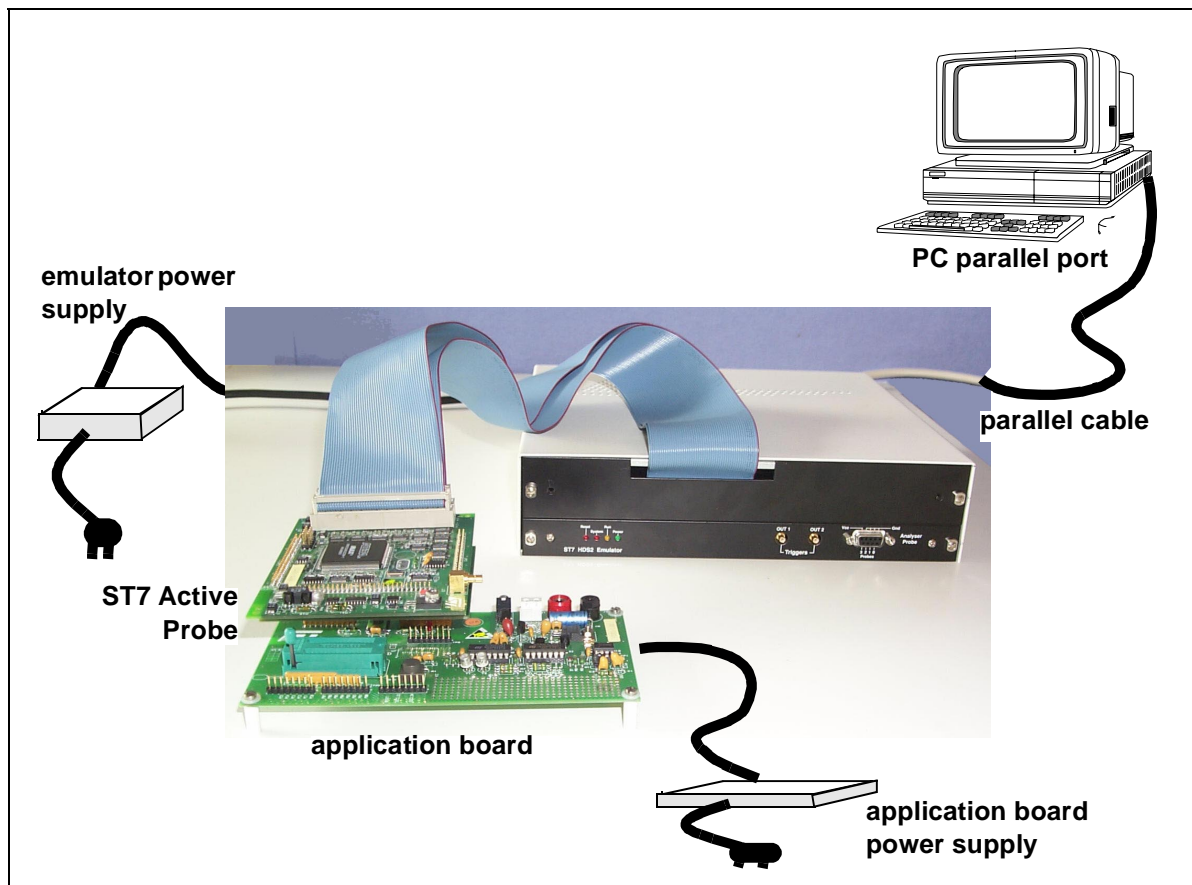
**Figure 1: ST7MDT6-EMU2B General Configuration**

## 1.2        Emulator Operation

A symbolic debugger, **ST7 Visual Debug**, (also referred to as **STVD7**), is provided to control the emulator.

ST7 Visual Debug can be run on a PC under the Windows environment, and is common to all ST7 devices. ST7 Visual Debug uses a window menu-driven interface, and enables you to configure the emulator.

*Chapter 3: STVD7* on page 17, explains how to install ST7 Visual Debug on your PC, and set up the emulator configuration so that you can begin your debugging session.

Once assembled and linked, the application software is ready to be downloaded into the ST7 emulator. The development station performs a real-time emulation of the target device, thus allowing high performance testing and debugging of both application hardware and software.

When the program is fully debugged, the ST7 EPROM programming board (ref.: ST7MDT6-EPB2 —not provided with this emulator kit) can be used to program the emulation device with the Motorola S Record format file produced by the OBSEND formatter.

## 1.3    Software and Documentation for the Emulator Kit

The "MCU on CD" CD-ROM contains:

- ST7 Tools, comprising the following software:

    - The source-level graphic debugger, ST7 Visual Debug, that operates with ST7-HDS2 Emulators and ST7 Development Kits or as a standalone ST7 simulator.

    - The ST7 Assembly chain, composed of an assembler, linker, librarian and formatter.

    - The ST7 Windows Epromer to program your MCU target devices.

- Third-party C compiler and toolchain demos (Hiware and Cosmic).

- ST7 application notes (with sources), training slides and exercises, this manual (in PDF version), and other useful reference materials.

- Datasheets for the ST7 MCU family.

## 1.4    About this manual....

Detailed instructions on how to install your emulator configuration is described in *Chapter 2: Getting Started* on page 9.

How to start debugging your application using your emulator and ST7 Visual Debug is described in *Chapter 3: STVD7* on page 17.

The emulator kit's hardware features are described in *Chapter 4: Emulator Features* on page 41.

## 1.5        Related Documentation

To get all the essential information about your ST7 MCU and the software that comes on the CD-ROM with the emulator kit, you will need to refer to these documents (also contained on the CD-ROM):

- ST7-Family Data Sheets

- ***ST7-Family 8-bit MCUs Product Overview*** (Ref. BKST7/2)

- ***Software Tools for the ST7 Family*** (Ref. Doc-ST7ASMLK-SW)

- ST7-Family Programming Manual

## 1.6        Getting Assistance

For more information, application notes, FAQs and software updates on all the ST microcontroller families, check out the CD-ROM or our website:

**http://mcu.st.com**

For assistance on all ST microcontroller subjects, or if you need help with using your emulator, use the contact list provided in *Product Support* on page 59. We'll be glad to help you!

# 2    GETTING STARTED

## 2.1    Your System Requirements

The ST7MDT1-DVP2 Development Kit (both hardware and software components) has been designed to work with PCs meeting the following requirements:

- One of the following operating systems: Microsoft® Windows® 95, 98 or NT®.

- Intel® Pentium (or compatible) processor with minimum speed of 100 MHz.

- Minimum RAM of 32 MB.

- 21 MB of free hard disk space to install all of the ST7 tools.

## 2.2    Delivery Checklist

The emulator kit (ref.: ST7MDT6-EMU2B) is delivered with:

- One emulator box containing the ST7 HDS2 main board (ref.: MB176).

- One parallel cable.

- Two 50-wire flat cables to connect the ST7 HDS2 main board to the emulation probe.

- One emulation probe—the ST7MDT6-Active Probe (ref.: DB397).

- Two device adapters—the SO34 (ref.: DB406) and SDIP32 (ref.: DB405).

- One User Manual for the ST7 Family Software Development Tools (assembler, linker, formatter, ST7 Visual Debug).

- This manual.

- One CD-ROM which contains ST7 information and ST7 toolchain software which includes ST7 Visual Debug.

An illustration of the main components of the emulator kit is shown in *Figure 2*.
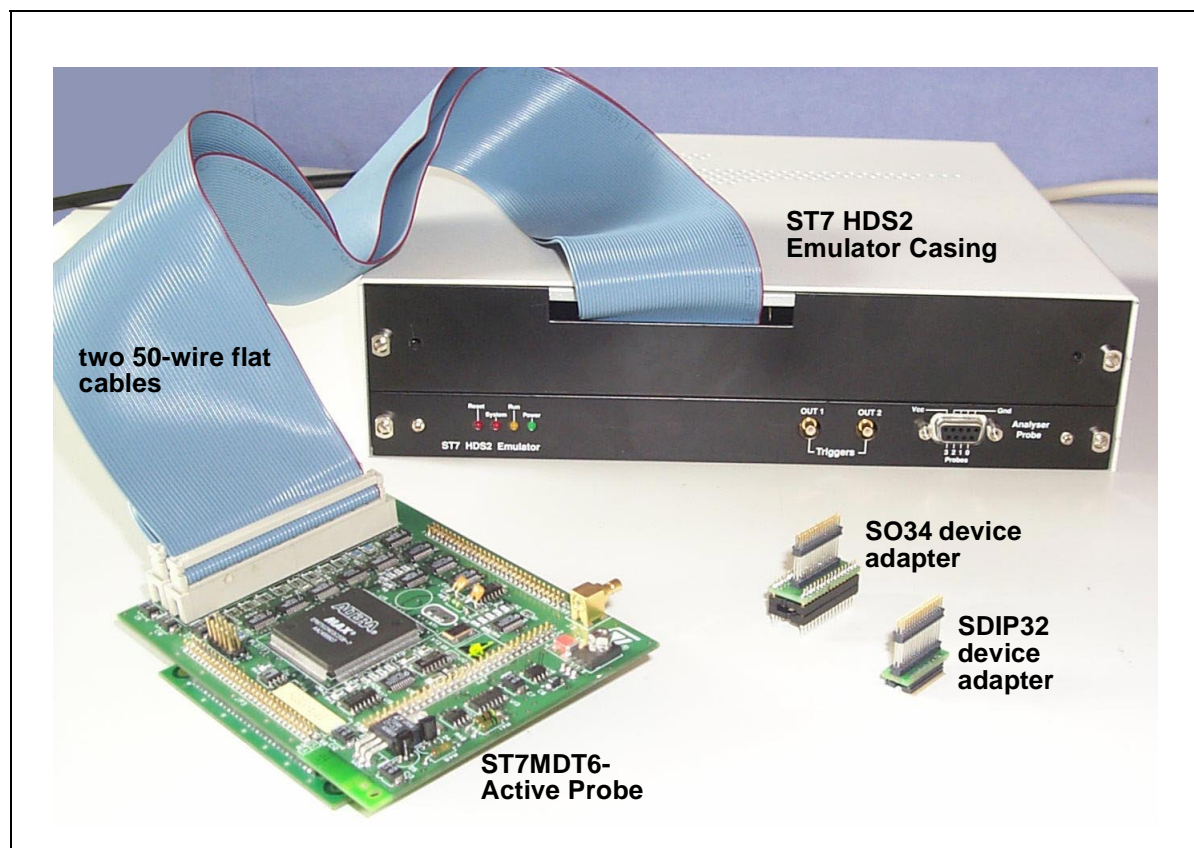


**Figure 2: Main Components of Emulator Kit**

## 2.3        Installing the Hardware

The ST7 HDS2 emulator is connected through the parallel port to a PC computer which runs the control software (ST7 Visual Debug) as explained later. To connect your ST7 HDS2 emulator, you will have to follow these general steps:

**1**   Connect the ST7 HDS2 to your PC using the parallel cable provided.

**2**   Connect the two flat cables of your ST7 HDS2 emulator to the emulation probe connectors.

**3**   Connect the appropriate device adapter (i.e. SO34 or SDIP32) to the emulation probe, then connect the device adapter to the matching socket on your application board.

**4**   Connect the power supply cable between the power supply block and the power connector located on the rear panel of your ST7 HDS2 emulator.

**5**   Power up the emulator and then connect your application power supply.

A connection flow diagram is shown below. Each installation step is described in detail in the following sections.
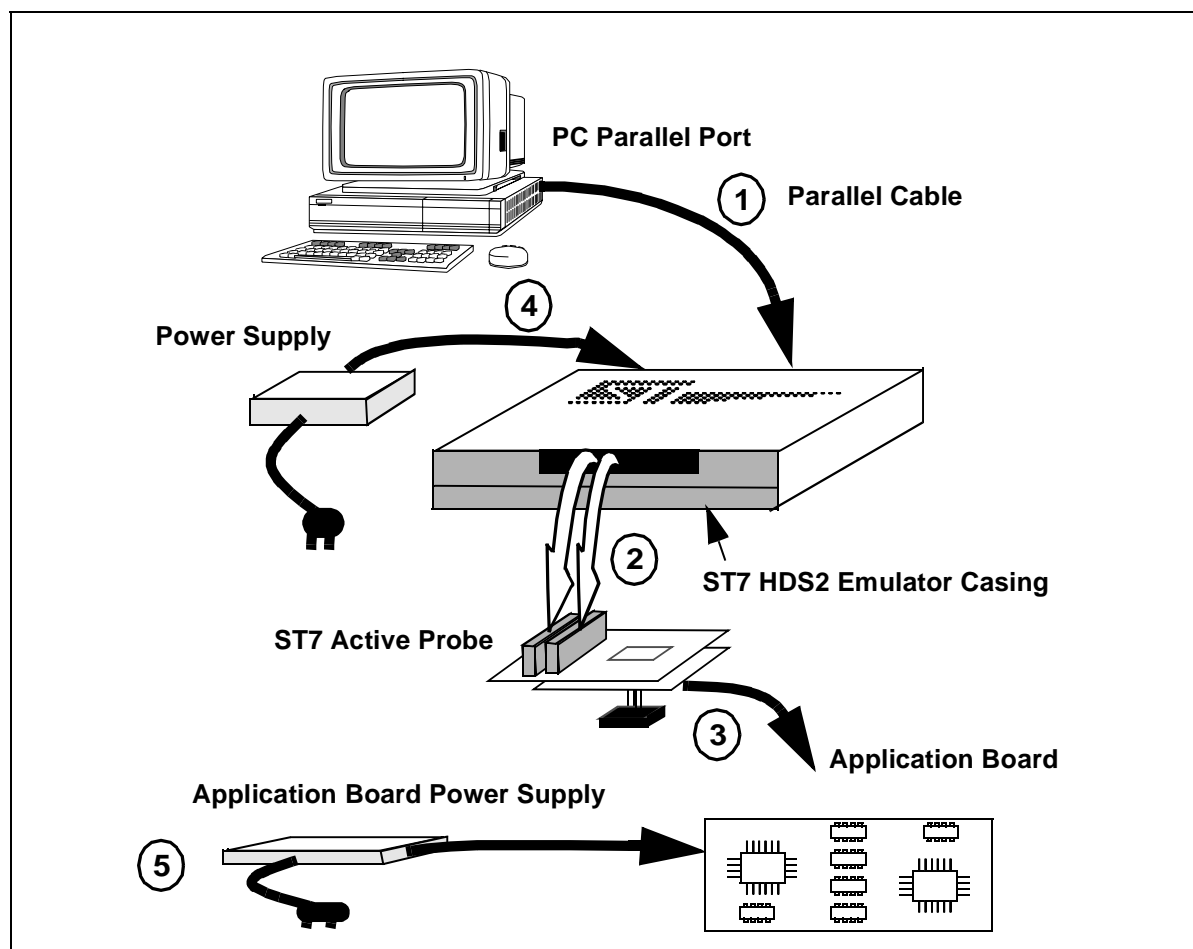


### Step 1: Connecting the Emulator to your PC

  **1**  Shutdown and power-off the PC that is to be connected to the emulator.

  **2**  Connect one end of the parallel cable to the emulator's rear panel 25-pin SUB-D connector and the other end to one of the PC's parallel ports (LPT1 to LPT2)—refer to *Figure 3*.

*Note:*  *Centronics (or PC-AT or SPP), ECP and EPP parallel port configurations are supported by the emulator.*

*Be sure to use the parallel cable provided with the emulator—using a longer parallel cable may cause emulator malfunctions.*
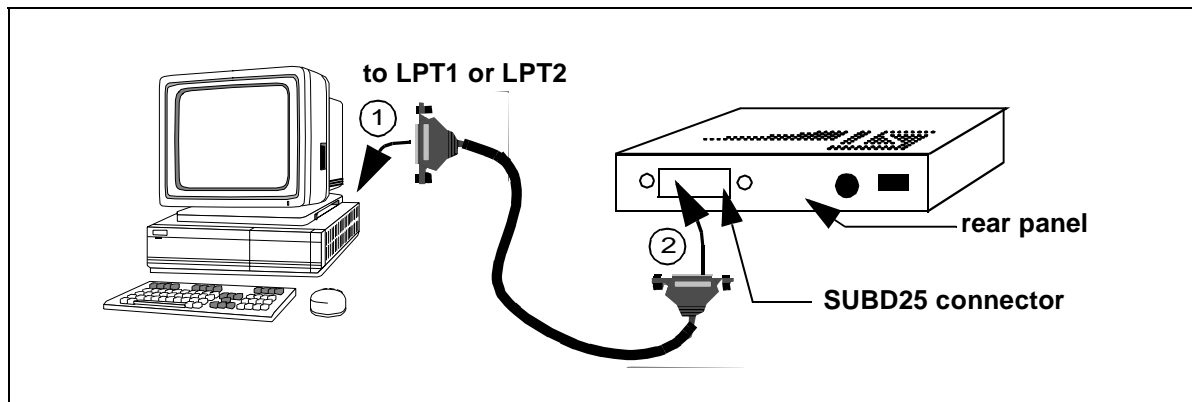


**Figure 3: Connecting the Emulator to the PC**

## Step 2: Connecting the HDS2 and the Probe

**1** Ensure that the application and the emulator are **powered-off**.

**2** Plug the two 50-wire flat cables into J1 and J2 as described below:

| ST7 HDS2 emulator (ref.: MB176) | ST7MDT6-Active Probe (ref.: DB397) |
|---|---|
| **Upper cable** | J2 |
| **Lower cable** | J1 |
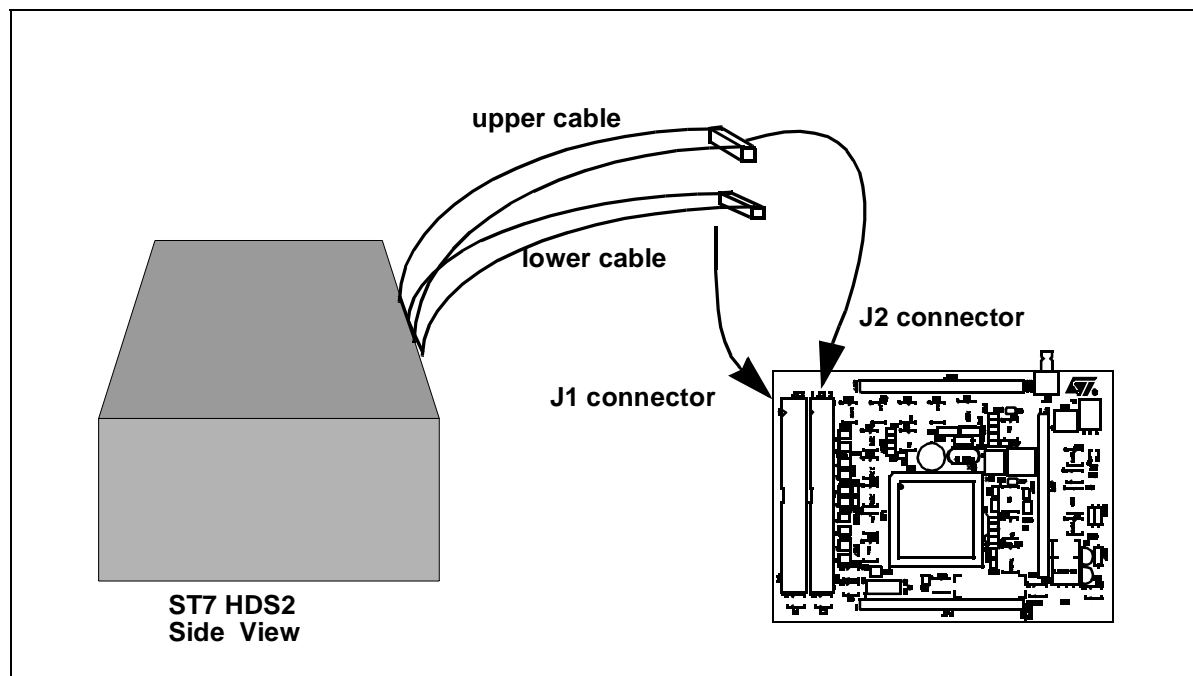
**Figure 4: Connecting the Emulator to the Probe**

**3  EMC-Compliant Probes:** In order to work under an EMC-compliant environment, you will have to clip one or two EMC-ferrite on each 50-wire flat cable linking the probe to the emulator box. Place these ferrites as close to the

emulator window as possible. Three ferrites are provided in the package. See *Figure 5* for an illustration of where to attach the ferrites.



**Figure 5: Making your Probes EMC-Compliant**

**Step 3:  Connecting the Probe to your Application Board using Device Adapters**

Use one of the two device adapters included in the emulator kit to connect the ST7MDT6-Active Probe to your application board.

If you are using the SO34 package, follow these steps (see *Figure 6*):

**1**   Solder the SO34 device adapter (ref.: DB406) onto your application board.

**2**  Now connect the ST7MDT6-Active Probe onto the SO34 device adapter by inserting the probe's female connector into the male pins on the device adapter. The number 1 pins of these two boards must also correspond.
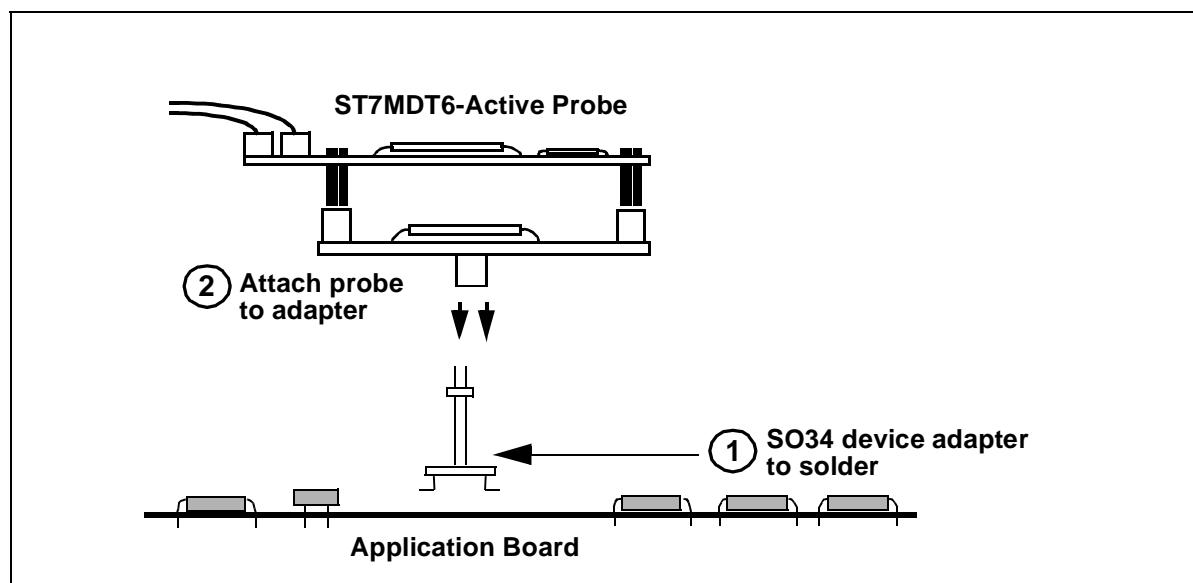


**Figure 6: Connecting the probe to your application**

If you are using the SDIP32 package, follow these steps:
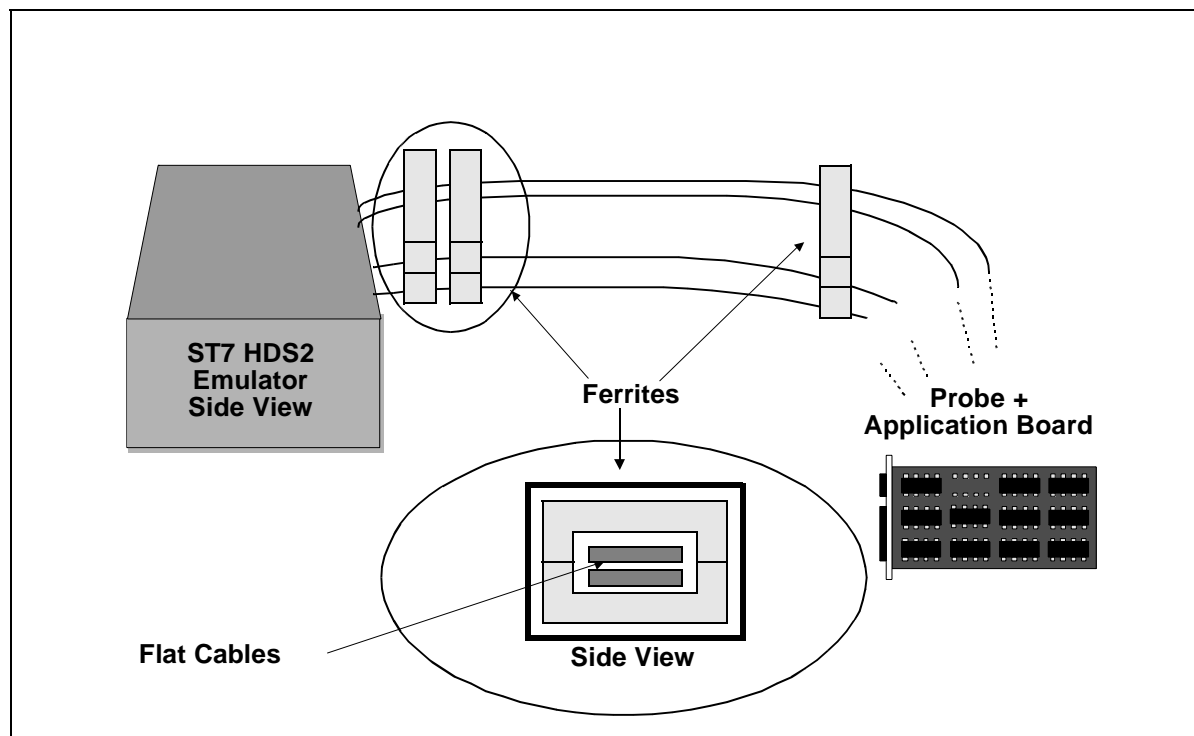
**1**  Connect the ST7MDT6-Active Probe onto the SDIP32 device adapter (ref.: DB405) by inserting the probe's female connector into the male pins on the DB405. The number 1 pins of these two boards must correspond.

**2**  Now connect the ST7MDT6-Active Probe and the SDIP32 device adapter onto your application board. The number 1 pins must also correspond.

**Step 4: Connecting the emulator power supply**

**Warning:**   **Make sure that both the ST7MDT6-EMU2B and the application are OFF before making any connections.**

**3**  Connect the external power supply provided with the emulator to the rear panel of the mainframe using a 5-pin DIN connector.

**4**   Plug the power supply into the mains using the supply cable provided.
Mains voltage specifications

| AC Voltage | 100 V to 240 V |
|------------|----------------|
| Frequecy   | 50 to 60 Hz    |

## Step 5: Powering up

**5**   Check the ST7 HDS2 operating voltage (110 V/220 V) indicated on the label on the power supply. Contact your dealer if this voltage does not correspond to your mains' power supply.

**6**   Power up ST7 HDS2 from the ON/OFF switch located on the rear panel. The LED labelled *Power* on the front panel should then light up.

**7**   Power up your application.

# 3      STVD7

STVD7 is an integrated development environment that allows you to edit, debug and rebuild your application all from within STVD7.

The following sections tell you:

- *Section 3.1*—how to install the STVD7 software,
- *Section 3.2*—how to launch STVD7,
- *Section 3.3*—a little about STVD7's debugging features,
- *Section 3.4*—what a workspace is,
- *Section 3.5*—what toolchains and application files are supported by STVD7,
- *Section 3.6*—how to create a STVD7 workspace,
- *Section 3.7*—how to open existing workspaces,
- *Section 3.8*—how to open binary files,
- *Section 3.9*—how to change your project settings,
- *Section 3.10*—how to save workspaces,
- *Section 3.11*—how to switch from the build context to the debug context,
- *Section 3.12*—how to configure the target MCU in order to debug more accurately and efficiently.

## 3.1      Installing STVD7

Your emulator comes with the *MCU on CD* CD-ROM which contains a number of ST7 software tools. These tools run under the Windows® 95, 98 and Windows NT® operating systems.

To install and setup the ST7 software tools, follow these steps:

**1**    Close all other open applications on your Windows desktop.

**2**    Insert the *MCU on CD* into your CD-ROM drive. The CD-ROM's autorun feature will open up a welcome screen on your PC. If the autorun feature does not work, use Windows® Explorer to browse to the CD-ROM' s root folder, and double-click on `Welcome.exe`.

**3**    Select ***Install Your Development Tools*** from the list of options. A new screen will appear listing the different families of STMicroelectronics MCUs.

**4**    Use your mouse to place the cursor over the ***ST7 Tools*** option. Choose ***ST Tools***, then ***ST7 Toolchain*** from the lists that appear.

**5**    The install wizard will be launched. Follow the instructions that appear on the screen.

You can choose to install the complete toolchain (i.e. the appropriate version of STVD7, the Windows Epromer and the Assembler-Linker) for each type of development tool (Development Kit, HDS2 or EMU3 emulators or simulator), or perform a customized installation.

If you choose a customized installation, you can choose to install any or all of the STVD7 versions, and/or the Windows Epromer and/or the Assembler-Linker.

*Note:*     *As a minimum, in order to use your emulator, you must install STVD7 for HDS2.*

If you also install the ST7 Assembly Toolchain, you will be able to use the ST7 Assembly Toolchain as part of STVD7's integrated development environment.
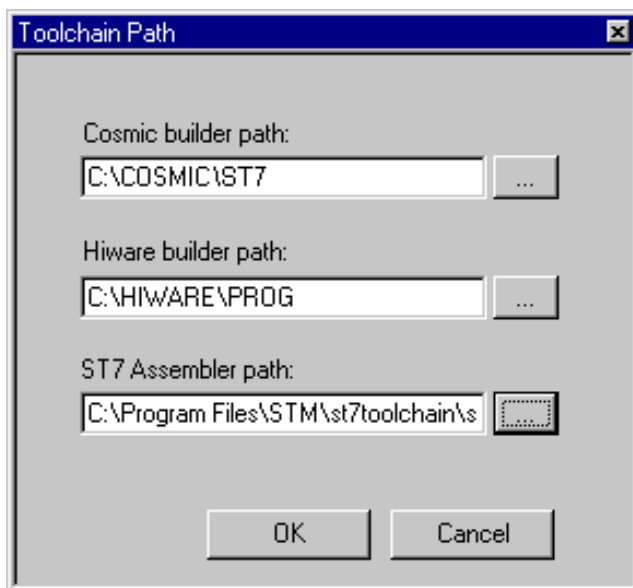
The installation is now complete. You will be prompted to reboot your computer. You should do so before launching STVD7.

## 3.2     Launching STVD7

**1** From your Windows desktop, select **Start>Programs>ST7 Tool Chain>Development Tools>STVD7 HDS2 emulator.**

**2** The first time you open a version of STVD7 you will be prompted to enter the toolchain paths to be used by STVD7's integrated development environment.

Enter the paths for the toolchains that you use (i.e. any or all of the Hiware, Cosmic or ST7 ASM toolchains) and click **OK**. (The default paths for each toolchain are shown below.)

**Toolchain Path**

Cosmic builder path:
`C:\COSMIC\ST7`                    [...]

Hiware builder path:
`C:\HIWARE\PROG`                   [...]

ST7 Assembler path:
`C:\Program Files\STM\st7toolchain\s`   [....]

OK          Cancel

**3** If you choose **Cancel,** you will be prompted again to enter the toolchain paths the next time you launch STVD7.

*Note:*     *You may modify the toolchain path at any time from within STVD7—simply select* **Project>Toolchain Paths** *from the main menu to access the dialog box above.*

**3.3      About STVD7 debugging features**

A number of advanced features are included in the STVD7 software:

- **Data Breakpoints** on the occurrence of a memory access via a read operation or a write operation, or both.

- **Instruction Breakpoints** on the occurrence of an opcode fetch.

- A **Logical Analyser** that allows you to control either the recording of the trace buffer, or a break in the execution of the application using a series of specific conditions (events).

- A **Trace window** to view the contents of the **trace buffer**, which permanently records in real time on 32-bits:

    - Address and data bus information.

    - Flag status and 4 external signal values.

    You can record up to 1024 executed cycles. Using trace filtering, you can filter out only those cycles you wish to record in the trace buffer. You can equally control which of the recorded cycles are displayed in the Trace window using line filtering. Addresses, data, control/status bits and 4 user signals are displayed using mnemonic and user symbols.

- Internal synchronization signals can be output to either of two **Trigger Outlets** on the front panel of the emulator (OUT1 or OUT2). This feature enables you to count events using an external equipment, when optimizing software for example, or to synchronize an oscilloscope when debugging hardware.

- **Hardware Events** can be used to control the sending of signals to the trigger outputs.

- You can choose the output that you wish the signal to be sent to (i.e. either OUT1 or OUT2).

- A **Hardware Test** function that allows you to perform a number of hardware tests on the Development Board, at your choosing. Refer to *Running the Hardware Test* for more information.

- A powerful **online help** facility can be invoked at any time to give additional information about the commands, the processor or the emulator kit.

**3.4          Workspaces**

STVD7 organizes project development and debugging into workspaces. Workspaces allow you to store application and project settings and save them as a `*.wsp` file, so that each time you wish to work on the project, you will find all of the settings exactly as you left them.

Creating a workspace is the first thing that you need to do when using STVD7 for the first time or when starting any new project. You must have an open workspace to work with STVD7. How to create a new workspace is described in detail in *Section 3.6* on page 24. Sample workspaces for each supported toolchain are provided so that you can familiarize yourself with STVD7 (for a listing of sample workspaces, see *Table 1* on page 22).

Each workspace is comprised of three information sets: the project settings, the visual environment and the debugging context.

• The **project settings** consists of the information necessary for a successful build of an application (commands to run, makefile file etc....). Your workspace's project settings include the definition of your application toolchain (see *Section 3.5* on page 21).

• The **visual environment** consists of the open windows elements along with their current layout, bookmarks and other features.The visual environment is composed of two environments, one in the **Build context** and one in the **Debug context** (see *Section 3.11* on page 33).

• The **debugging information** includes information on breakpoints, memory mapping, advanced breakpoints programs, trace etc...

**3.5       Toolchains and application files**

A quick summary of development toolchains and application file types supported by STVD7 will help you in setting up your workspace.

Three different development toolchains are currently supported by the STVD7. Each type of toolchain has its own application file types, project environment and building tools (i.e. linkers and convertors):

- The **ST7 macroassembler toolchain** from STMicroelectronics, which generates either `.s19` or `.hex`  application files with various intermediate files, such as `.map` or `.lst` files.

- The **Hiware C or Assembler toolchain**, which generates `.abs` application files with various intermediate files, such as `.o` or `.dbg` files.

- The **Cosmic C or Assembler toolchain** which generates `.elf` application files with various intermediate files, such as `.o` or `.st7` files.

When you set up a workspace, you will need to define the following project settings:

- **The toolchain to be used—**Hiware, Cosmic or ST7 macroassembler.

- **The executable application file** (`*.abs`, `*.elf`, `*.s19` or `*.hex` depending on toolchain—refer to *Table 2* on page 23).

- **The maker program** for the toolchain. The maker program can be a part of the toolchain software (such as Hiware's `maker.exe`) or you can choose to use a generic maker such as `Nmake.exe` or `Gmake.exe` (which is provided with the STVD7).

- **The maker batch file** (`*.mak` or `*.bat`). This is a file which you create for each application which spawns the compilation and/or link step each time you wish to build or rebuild. In it, you define the conditions for recompiling, re-linking or both.
  Default `*.mak` or `*.bat` files are often included with the toolchains—for example, maker.mak is included with the Hiware toolchain and simply recompiles your application if it detects that the file has been saved since the start of your debugging session. The STVD7 software includes sample `*.mak` and/or `*.bat` files for each toolchain—these are listed in *Table 1*.

**Table 1: Sample files included with STVD7**

| Toolchain | Sample Workspace (with default path | Sample Make and/or Batch files (with default path[1]) | Description of Make/Batch File |
|---|---|---|---|
| **ST Macro assembler** | `.../realtim/realtim.wsp` | `.../realtim/tim_rtc.bat` | Batch file that forces a recompile of application file. |
| | `.../spim11/spim11.wsp` | `.../spim11/spim11.bat` | Batch file that forces a recompile of application file. |
| **Cosmic** | `.../c/cosmic/sample.wsp` | `.../c/cosmic/sample.mak` | Recompiles only if the application file has been resaved. |
| | | `.../c/cosmic/sample.bat` | Forces a recompile of application file. |
| **Hiware** | `.../c/hiware/sample.wsp` | `.../c/hiware/build.mak` | Recompiles only if the application file has been resaved. |
| | | `.../c/hiware/rebuild.mak` | Forces a recompile of application file. |

1) The full default path is: `C:/Program Files/Stm/st7toolchain/stvd7/hds/sample/...`

### 3.5.1    About application files

The user should verify that the options to include debug information were active during creation of the project files. *Table 2* on page 23 summarizes the way each toolchain functions and lists the different file types (source files, intermediate files and application files) used and produced by the toolchain. The **application file types** and **intermediate file types** necessary to exploit fully the STVD7 capabilities are listed.
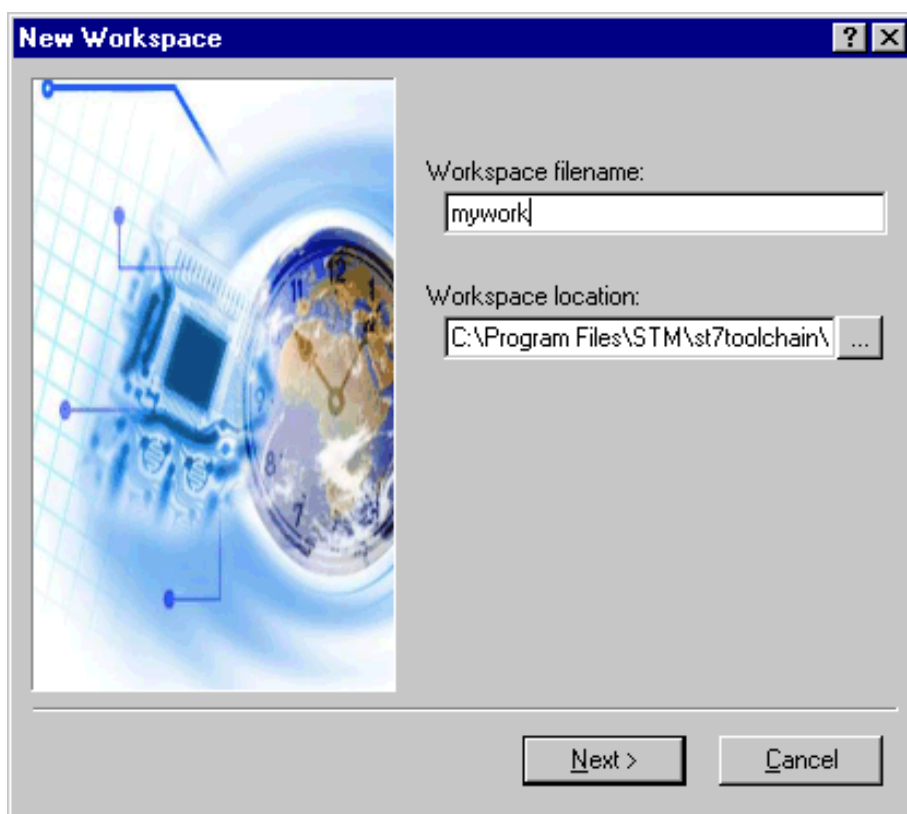
**Table 2: Toolchain steps and their output files**

| Toolchain: | ST Macroassembler | Hiware | Cosmic |
|---|---|---|---|
| **Compile or Assemble Step:** | | | |
| Source File Types | `.asm` | `.c, .asm` | `.c, .s` |
| Required Options | `asm –li macrost7.asm` | `<NONE>` | `+debug` |
| Resulting File Types | `.obj, .lst` | `.o, .dbg` | `.o` |
| **Linker Step:** | | | |
| Required Options | `lyn macrost7.obj, macrost7`<br>`asm macrost7.asm  –sym –fi=macrost7.map` | `<NONE>` | `<NONE>` |
| Resulting File Types | `.map, .lst` | `.abs` | `.st7` |
| **Converter Step:** | `obsend macrost7, f, macrost7.s19, srec`<br>or<br>`obsend macrost7, f, macrost7.hex, intel` | not applicable | `cvdwarf` |
| **Resulting Application File:** | `.s19 or .hex` | `.abs` | `.elf` |
| **Necessary Intermediate Files:** | `.map, .lst` | `.o, .dbg` | `<NONE>` |

The **application file(s)**, **source files** and any necessary **intermediate files** (these are listed above and contain debug information necessary to the STVD) should be located in the same project directory. You do this when you define your workspace.
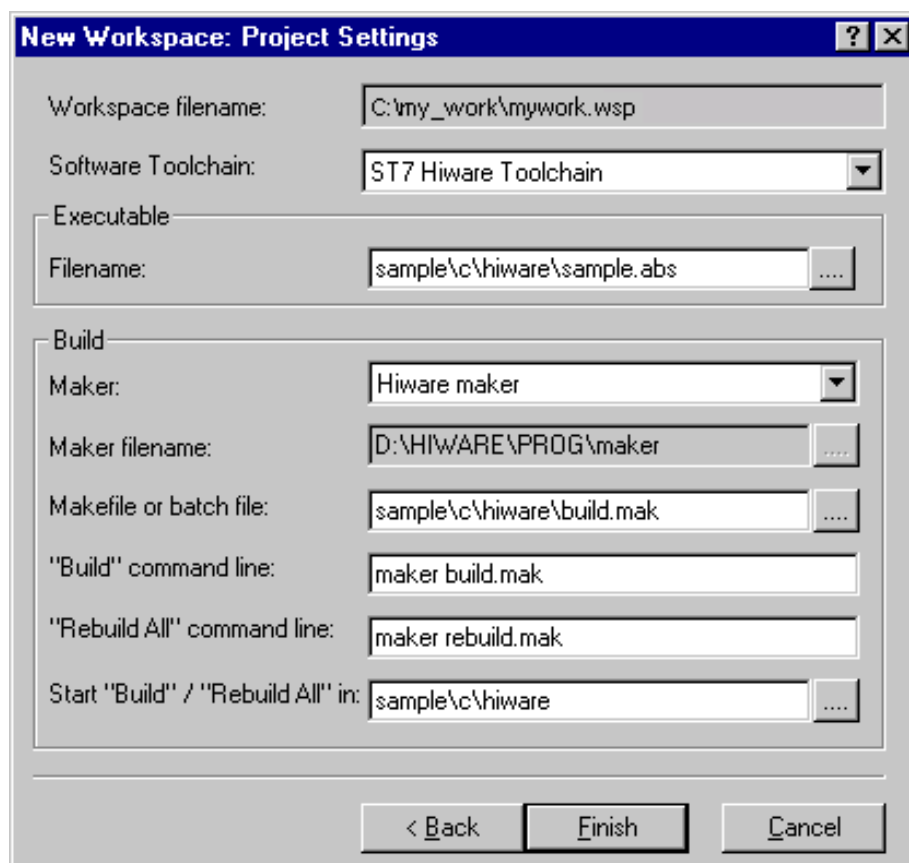
*Note:*    *It is always preferable to have access to all of the files generated by the development toolchain. However, you can load* `*.s19` *or* `*.hex` *binary files directly and have limited debugging capabilities (refer to Section 3.8 on page 28).*

## 3.6      Creating a workspace

**1**   Select *File>New Workspace*. This command opens a new window where you define the name of your workspace and the directory in which you want to work.

**2** Then, click *Next>*. The **New Workspace: Project Settings** dialog box appears:



Here you enter your software toolchain, your executable filename and your build parameters either by typing or using the drop boxes.

**3** Select the toolchain and enter the name of your application's executable file. For example, if you wish to use the Hiware toolchain for ST7, your executable file will be of type `*.abs` (refer to *Table 2* on page 23)—click on the browse

button `....` to browse to the folder where your executable file is saved and select it.

**4** Next, choose the type of maker your application uses from the drop down list. In the example above, we have chosen the default Hiware maker, `maker.exe`. STVD7 will automatically look for this maker file in the folder you defined as the Hiware toolchain path.

**5** Finally, you must define a make file or a batch file. There are several sample files provided with STVD7 (see *Table 1* on page 22). Here we have chosen

build.mak as the default make file, used when the **Build** command is issued, and rebuild.mak as the make file to use when the **Rebuild** command is issued.

**6**   After you have finished defining your project settings, click **Finish**.

Once the workspace is opened, the Workspace window displays its contents.

When you create a new workspace, the first time you switch to Debug context (see *Section 3.11* for an explanation of STVD7 contexts), the MCU Configuration window will automatically open to prompt you to choose you target MCU and confirm or modify its option and memory configuration (see *Section 3.12* on page 34).

## 3.7    Opening an existing workspace

If you have already created a workspace, you simply need to open it in order to load all of your project settings into the STVD7.

*Note:*    *There are a number of sample workspaces provided with STVD7 that you can open to get familiar with STVD7. These samples are listed in Table 1 on page 22.*

**1**   From the main menu, select **File>Open Workspace.**
    This command opens a window where you can browse to any folder you wish, and select an existing workspace.
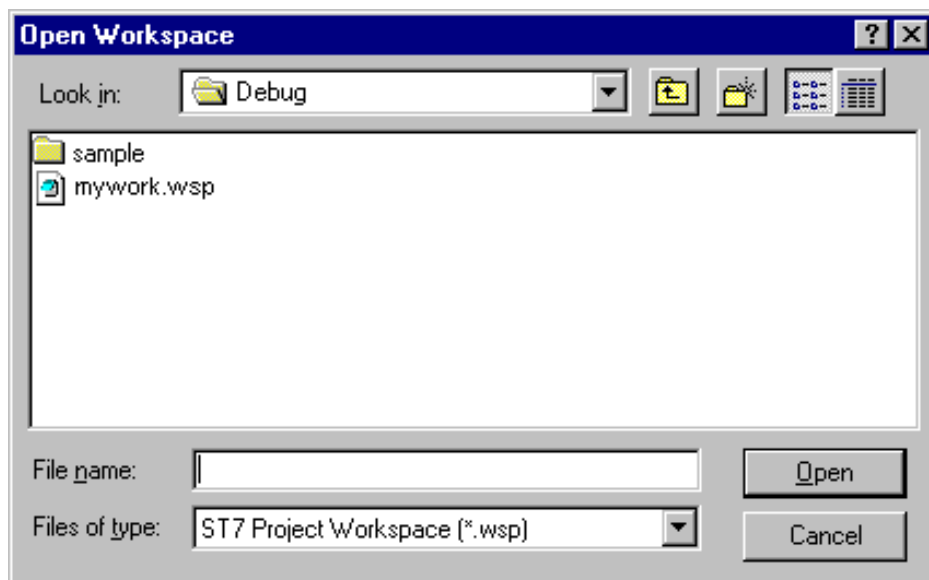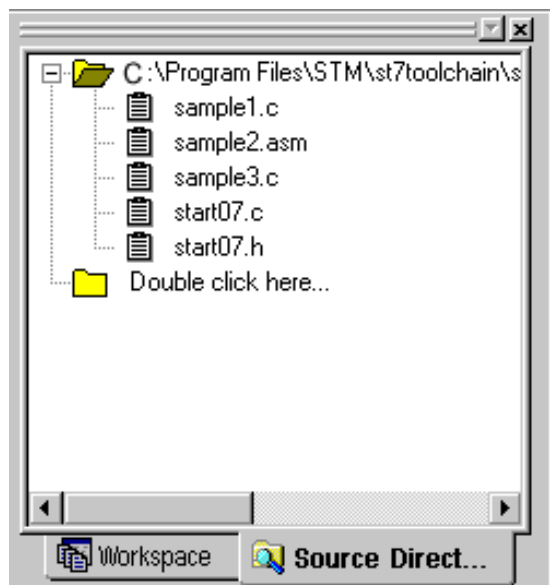
**2** The Workspace window opens.

When a workspace is opened, all of the predefined project settings are loaded into the STVD7. The **Workspace** window will show a structured representation of the project. For example, mywork.wsp shows that it uses build.mak as the make file and sample.abs as the application file.
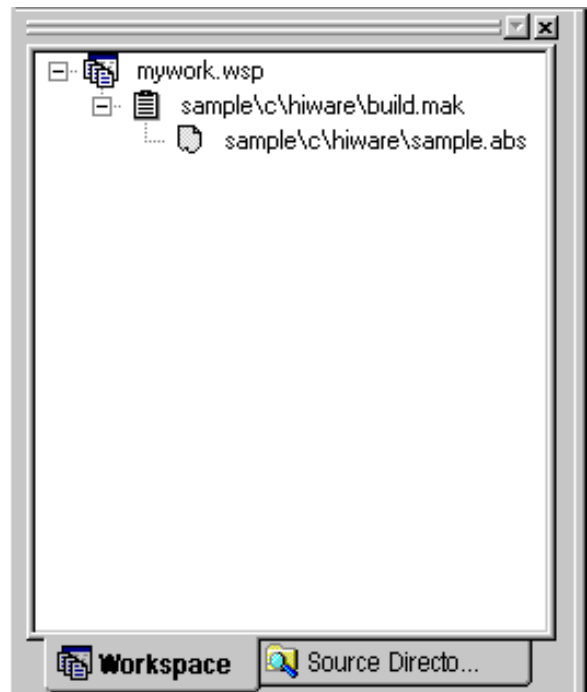


*Note:* *Although the name of the application file is shown in the Workspace window, it has not yet been loaded into the emulation memory—see page 28.*

If you click on the **Source Directory** tab, the window will show every source and intermediate file type (\*.c, \*.s, \*.asm, \*.h or \*.o) in the selected directory.



**3** If there are no source files shown in the Source Directory tab of the Workspace window, or you wish to list additional files stored in another folder, you may browse to them by clicking the ***Double Click here...*** folder. The **Add Source**

**Directory** window pops up allowing you to enter or browse for a new directory, and filter out the file types of interest.
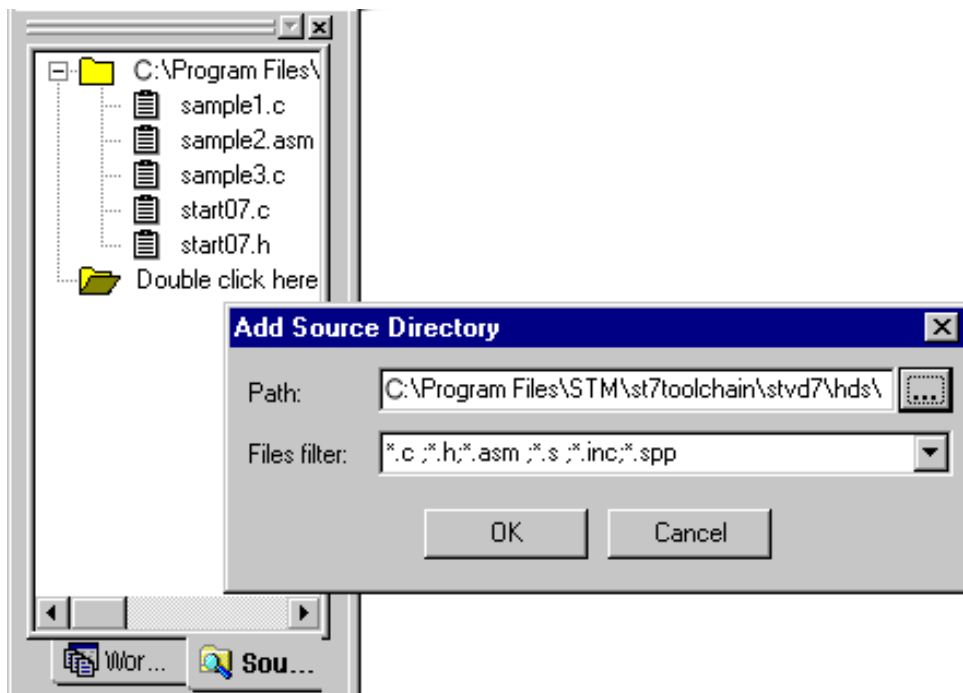


**4** To load the application file, as well as any intermediate files, click the Debug icon or the Reset Chip icon. The application and symbols will be loaded. Before you can start debugging, you must set the target hardware device by configuring the MCU.

## 3.8    Opening binary files

If you do not have access to the source or intermediate files generated by a toolchain, you may also load `.s19` and `.hex` files on their own using the **Open Workspace** command.

*Note:*    *The range of debugging features available when you open a binary file only will be very restricted. You will only have access to the Disassembly Window.*

**1** Launch STVD7 and select **File>Open Workspace** from the main menu.

**2** Browse to the folder where your binary file is stored, and select *__All files (\*.\*)__* in the *Files of type* field.
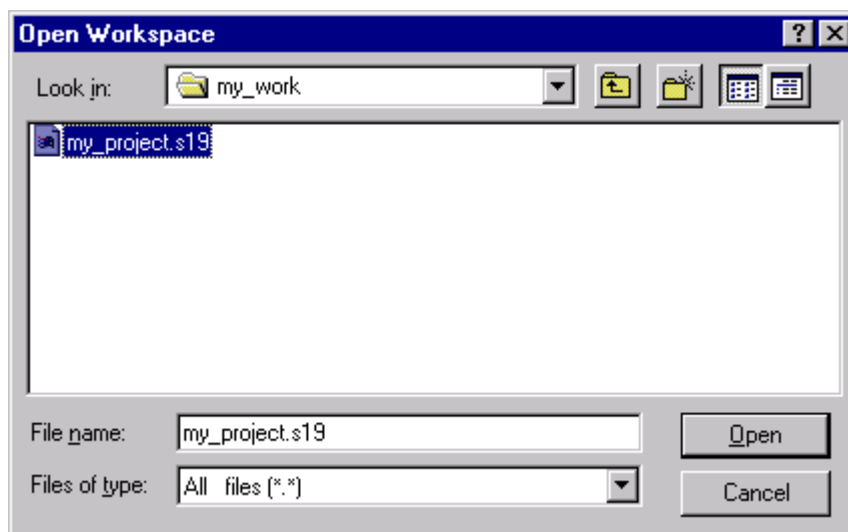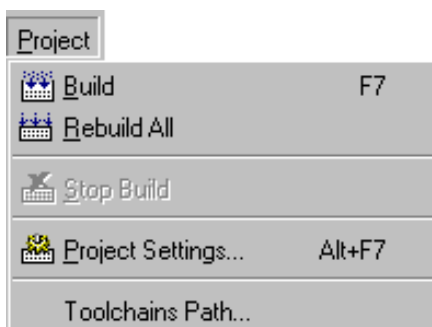


**3** Select your binary file (`*.hex` or `*.s19`) and click *__Open__*.
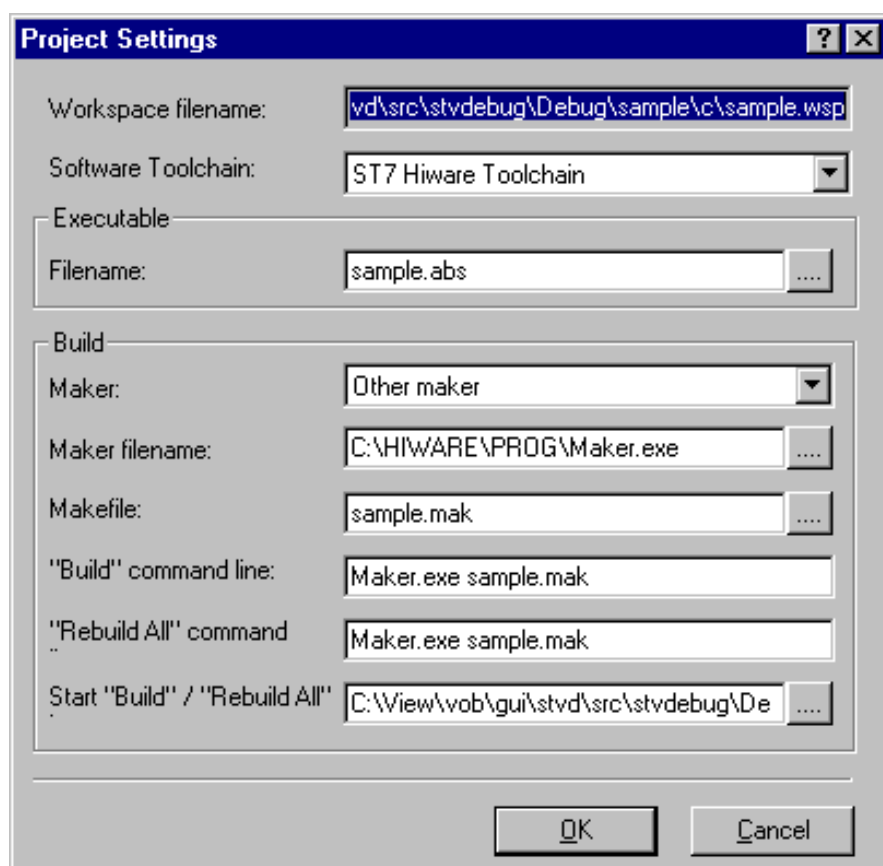
The binary code in the `.s19` or `.hex` file will be loaded into STVD7 and you will be able to access the Disassembly Window. A workspace file (of the same name as the binary file, but with an extension `.wsp`) will be created automatically.

## 3.9   Changing your project settings

The Project menu contains the *__Build__* and *__Rebuild All__* commands you need to recompile your application after having made changes to it in the course of debugging. You may also access your project or toolchain settings in the event you wish to change them.

From the main menu, select **Project>Project Settings**.

| Project Settings | ? X |
|---|---|
| Workspace filename: | vd\src\stvdebug\Debug\sample\c\sample.wsp |
| Software Toolchain: | ST7 Hiware Toolchain ▼ |

**Executable**

| Filename: | sample.abs | .... |
|---|---|---|

**Build**

| Maker: | Other maker ▼ |
|---|---|
| Maker filename: | C:\HIWARE\PROG\Maker.exe | .... |
| Makefile: | sample.mak | .... |
| "Build" command line: | Maker.exe sample.mak |
| "Rebuild All" command | Maker.exe sample.mak |
| Start "Build" / "Rebuild All" | C:\View\vob\gui\stvd\src\stvdebug\De | .... |

[ _OK_ ]   [ _Cancel_ ]

You can change your settings here and continue running your application. When you exit STVD7, the system will ask you if you want to save these settings in the workspace you have been working in. If you choose **yes**, these will become your new workspace settings; if you choose **no**, these settings will be lost.

The **Toolchain Path...** item invokes the following window:
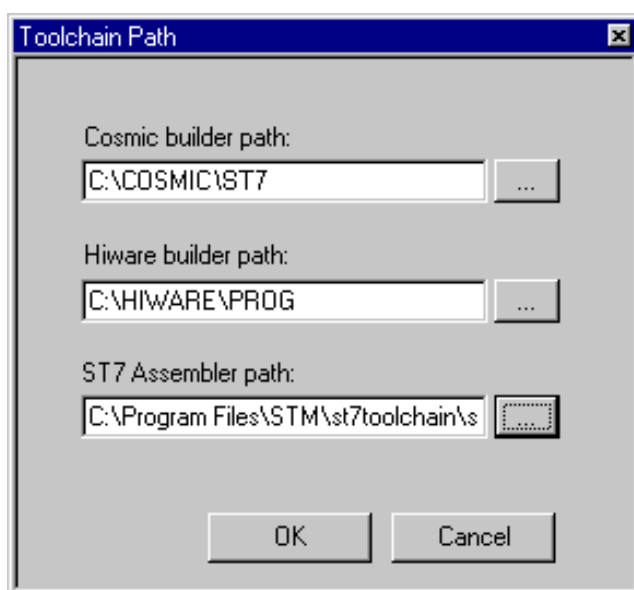


In this window, you can define your builder and/or Assembler paths. Clicking [...] opens a browser window.

### 3.10    Saving workspaces

Whenever the current workspace is closed, it is automatically saved. This can happen either when exiting STVD or opening or creating a new workspace.

In addition to this, a workspace can be explicitly saved with the **File>Save Workspace...** or **File>Save Workspace as...** commands.

The user is given the choice of which of the workspace elements to include in the saved file. Either the **visual environment** or the **debugging information** may be saved alone, or both may be saved together. This is configured as follows:

**1**    From the main menu, select **Tools>Options**.

**2**    In the Options window that opens (see *Figure 7 on page 32*), select the **Workspace** tab.

**3**    Choose whether you wish your saved workspace to include either the visual environment or the debugging information or both.

**4**   Select which windows will appear docked when a project is opened by checking the appropriate check boxes in the *Floating windows in the main frame* area. Only windows currently docked in the main window can be included.
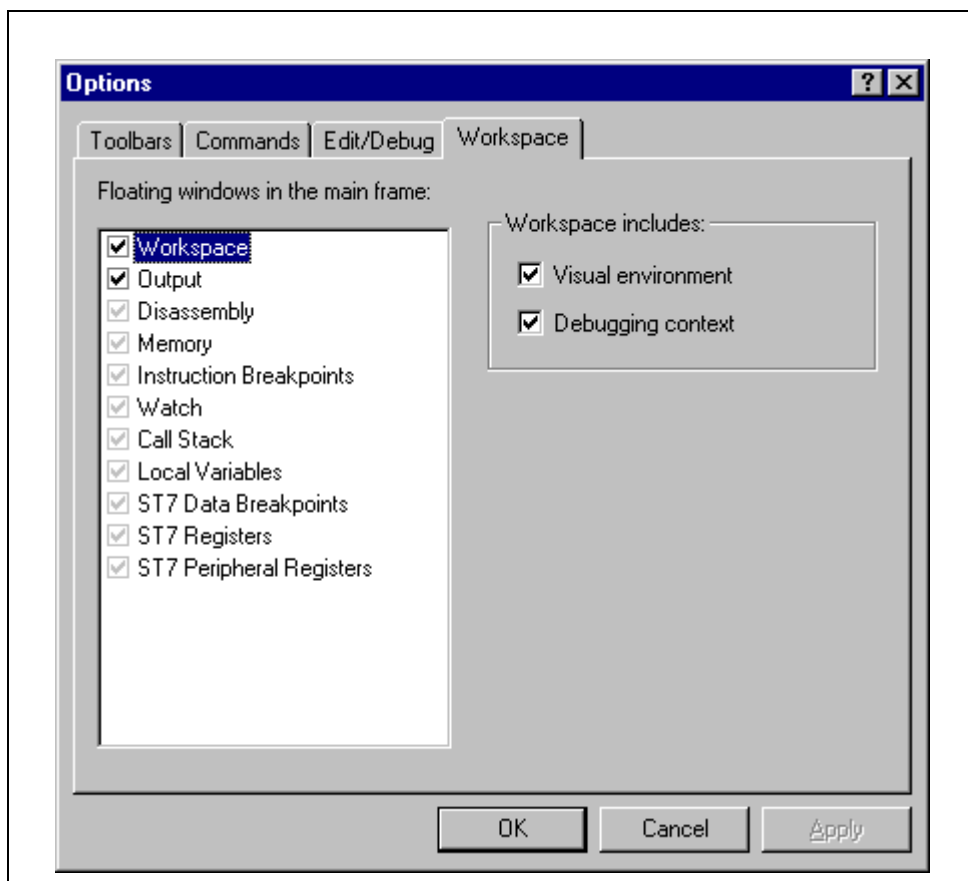


**Figure 7: Options window**

**1**   Click ***Apply*** to confirm your settings.

**2**   Click ***OK*** close the dialog box.

In addition, open file contexts and current window positions are saved when the workspace is closed. This feature restores the workspace window, window layout and file views to that which was current when STVD7 was closed. The toolbar layout, plus customized toolbar content is also saved and restored with the workspace (options set via the tabs entitled **Toolbars** and **Commands**).

By default (i.e. when saved automatically) the workspace is saved as file *<application>*.wsp. The name of the file corresponds to the name used for the executable file (for example, *<application>*.abs for a Hiware application file).

*Note:*          *Using the **Configuration Setup** dialog box (available from the MCU Configuration dialog box), you can also control what type of MCU configuration information is restored from a workspace file (\*.wsp).*

## 3.11      Debug context and Build context

There are two STVD7 contexts, the **build context** and the **debug context**. Until now, in creating a workspace, and defining your project settings, you have been in the build context. To proceed step—configuring your MCU—you need to change to the debug context.

Briefly, the two contexts are different in that:

- In the build context, you can open and close workspaces and build or re-build the application executable file.

- In the debug context you set the emulated MCU configuration (this step is described in *Section 3.12* on page 34) and debug the executable file created while in the build context.

### 3.11.1      Build Context

The build context is the context set when starting STVD7. In this context, it is not necessary to be connected to an emulator and the debug commands are not available. You can also edit the source files of an application and perform the use the **Build** command to perform compile and link actions in an interactive and iterative way to re-build the application executable file.

### 3.11.2      Debug Context

In this context, the following debug actions can be carried out:

- Loading, running and stopping the application.

- Defining the MCU configuration (MCU options and memory mapping).

- Viewing source and disassembled code.

- Setting instruction breakpoints with a counter and/or condition.

- Setting data breakpoints.

- Viewing local variables, memory and ST7 registers.

- Viewing history of execution from the trace buffer or with the Call Stack feature analyzing the performance of a piece of code.

### 3.11.3    Switching between contexts

The switch between contexts usually occurs when the **Start Debugging** and **Stop Debugging** commands are used:

From the main menu, choose **Debug>Start Debugging** or **Stop Debugging** or click on the **Start Debugging** or **Stop Debugging** icons shown at right.

While debugging, the editor allows source files to be modified. To switch to the Build context perform either a **Build** or **Rebuild** action or use the **Stop Debugging** command.

### 3.12    Configuring the MCU

After you create or open a workspace, the next step you must perform before starting your STVD7 debugging session is to define and configure the target device (MCU) that you wish to emulate.

The target device is defined and configured from the MCU Configuration window.

**1**    First, ensure that you are in **Debug context** by clicking on  . (STVD7 has two contexts: Debug context and Build context—these are described in *Section 3.11*.)

*Note:*        *The first time you enter into the Debug context after having created a new workspace, the MCU Configuration window will be opened automatically.*

**2**    Select **Tools>MCU Configuration** from the main menu. The MCU Configuration window will open.

An example of a typical MCU Configuration window is shown in *Figure 8*.
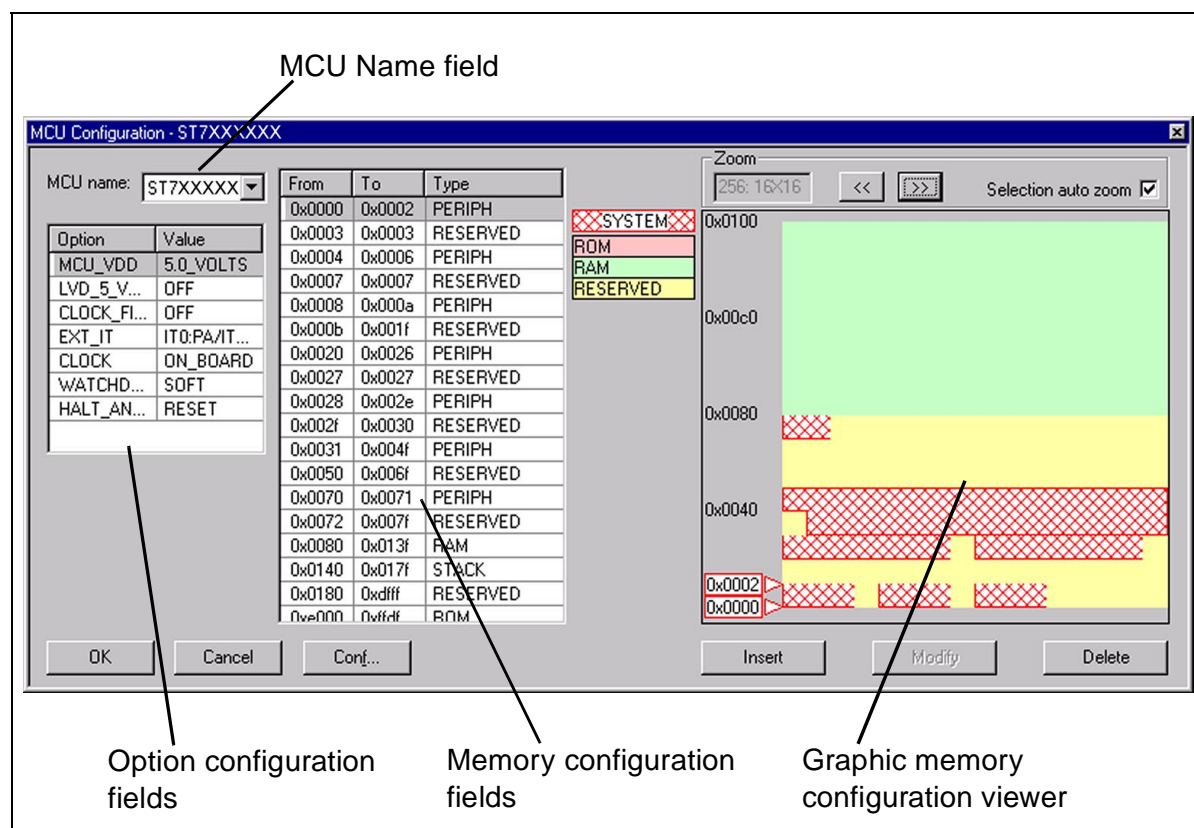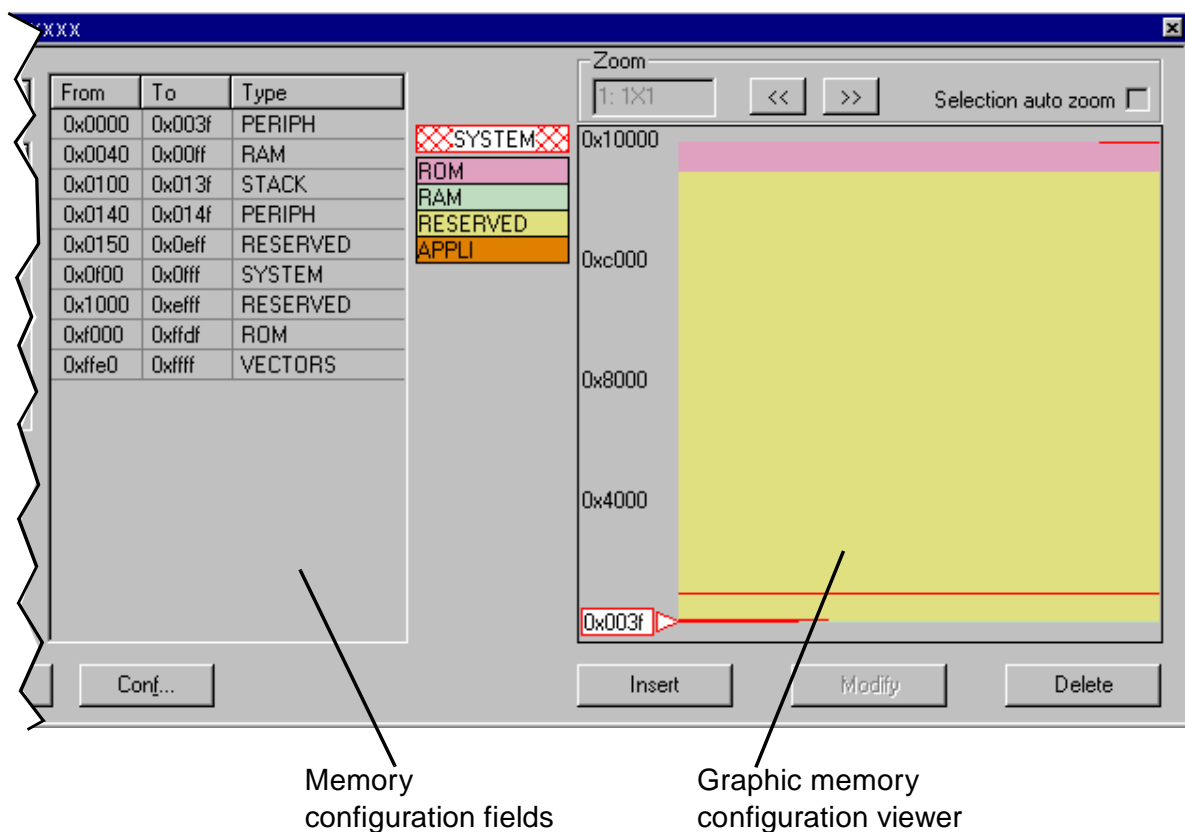


**Figure 8: MCU Configuration window**

*Note:*        *The options shown in the above example may not be available for your particular target MCU.*

   **3  Set the Target MCU**. In the MCU name field, select the target device for which
      the application is intended from the dropdown box. Once a target MCU has
      been chosen, the *Option configuration* and the *Memory configuration* fields will
      show the default values for this device.

   **4  Configure the MCU Options and On-Chip Peripherals.** All of the
      configurable options on your target hardware device are listed in the *Option
      configuration* fields. Beside each *option*, a default *value* is given. You may
      change this value by clicking on it and choosing a new value from the drop
      down list.   This allows you to configure your target device's options and on-chip
      peripherals. Depending on the MCU selected, the default settings in the *Option
      configuration* fields will change. It is up to you to configure those options that will
      impact your application so that the emulator accurately emulates your target
      device.

*Note:*      *For more information about the configurable options available on your target hardware device, please consult your target MCU's datasheet.*

**5 Configuring the MCU Memory.** The default memory settings depend on the MCU selected. However, you can configure the memory settings as you wish if your application requires non-default settings. This feature would enable you, for instance, to temporarily increase the ROM size during the development phase of your application.



| From | To | Type |
|------|------|---------|
| 0x0000 | 0x003f | PERIPH |
| 0x0040 | 0x00ff | RAM |
| 0x0100 | 0x013f | STACK |
| 0x0140 | 0x014f | PERIPH |
| 0x0150 | 0x0eff | RESERVED |
| 0x0f00 | 0x0fff | SYSTEM |
| 0x1000 | 0xefff | RESERVED |
| 0xf000 | 0xffdf | ROM |
| 0xffe0 | 0xffff | VECTORS |

Memory configuration fields

Graphic memory configuration viewer

There are two methods for configuring the memory settings on the MCU: by typing in the start and stop addresses of each memory zone into the **memory configuration window**, and by graphically moving the memory zone boundaries in the **graphic memory configuration viewer** (see *page 38* for more instruction).

**Memory zone types**

The left column of the **memory configuration window** indicates the address range of each memory zone. The right column indicates the memory type of each zone. Depending on your target MCU, the available memory types may be: Peripherals, RAM, ROM, Stack, System, EEPROM, Reserved, Vectors,

Application. Some of these zones can have their type and size modified, others cannot be modified. Their definitions and properties are explained as follows:

- **Peripherals**: Microcontroller internal or rebuilt peripherals registers. Their properties are defined as in the microcontroller user manual. This memory cannot be modified.

- **RAM**: Random-Access-Memory of the microcontroller. This memory type can be modified.

- **ROM**: Read-Only Memory of the microcontroller. Write protected. This memory type can be modified.

- **Stack**: Stack of the microcontroller. This memory type cannot be modified.

- **System**: The emulator uses this space for emulation management. This memory type cannot be modified.

- **EEPROM**: This memory is internal to the microcontroller and is located inside the emulation device. The programming of this zone is done according to an automaton found in the user manual. This memory type cannot be modified.

- **Reserved**: This memory zone is reserved as on the microcontroller. It is not allocated to any use and is write protected. This memory type cannot be modified.

- **Vectors**: This memory zone contains the user interrupt vectors zone. It is write protected. This memory type can be modified.

- **Application**: This memory type is microcontroller-specific. The user can add memory or peripheral resources on its hardware. It is not available on every emulator. Properties are linked to the user hardware. This memory type can be modified.

For most target MCUs, you may modify the following types of memory zone: **RAM**, **ROM**, **Reserved** and **Application**. This feature would enable you, for instance, to temporarily decrease the RAM zone, increase the size of the ROM (to exceed what is available on the real microcontroller) during the first stages of development. Once your program is functional, you can start to optimize its size by reducing your code and returning these zones to their original size. There are two different actions you may perform on the memory configuration:

- change the type of an entire existing zone.

- define a new zone of any type wherever possible.

**To change an existing memory zone:**

    **1**    Select the memory zone to be modified.

    **2**    Click on the *Modify* button at the bottom of the window. A **New MCU Memory Range** dialog box will open, allowing you to change either the address range and/or the memory type of the memory zone.
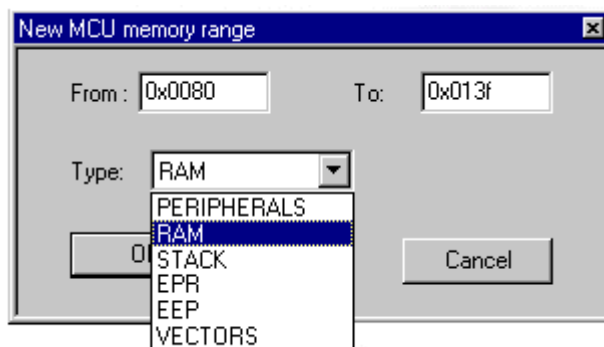


**To create a new zone of any type:**

    **1**    Click on the *Insert* button. The **New MCU Memory Range** dialog box will appear.

    **2**    Enter the address range of the new memory zone in the *From* and *To* fields.

    **3**    Select the type of the new memory zone in the *Type* field.

    **4**    Click *OK* to validate your choice.

The new memory zone will then appear in the **MCU Configuration** window **unless** you tried to create a new zone in a non-modifiable memory space (such as Stack or EEPROM).

**To use the Graphic Memory Configuration viewer:**

    **1**    In the memory configuration window, click on the zone whose boundaries you wish to move.

    **2**    Check the *Selection auto zoom* box in the upper right-hand corner. The graphical view of the memory configuration will be scaled so that the zone you have selected is easily visible.

    **3**    At the upper and lower boundary of the zone, at the left-hand side of the graphical viewer, you will see a small triangle and rectangular box giving the boundary addresses of the memory zone. You can change a boundary address by dragging and dropping the triangle with the mouse to its new location. The triangle can be moved either up or down, left or right in the graphical viewer.

The MCU configuration that you specify will, by default, be saved in a workspace file (*.wsp) for the project. The next time the application is opened, the STVD will automatically set the MCU configuration (as well as the layout of opened windows and other debug information) to the same conditions you had when you left the last debugging session.

If you do not wish your MCU configuration information to be saved in the workspace file, you must alter the default Configuration Setup options by clicking on the **Conf...** button.

### 3.13    Start debugging!

Once in **debug context**, you are now ready to start debugging your application using the emulator.Full documentation on how to:

- control your STVD7 work environment

- use its integrated editor

- use the many debugging windows and features

is available from the online help and the online STVD7 user manual, located under **Help** in the main menu.

# 4       EMULATOR FEATURES

## 4.1     Main Features of the ST7 HDS2 Emulator series

The features described below are common to all ST7 HDS2 emulators:

- Real-time emulation capability (internal frequency from internal 0.5 MHz up to 8 MHz).

- Full memory emulation (up to 64 KB).

- Real-time trace with 3 event conditions allowing selective recording.

- Hardware breakpoint capability on instruction Fetch.

- Hardware breakpoint capability on address.

- Breakpoint capability on invalid address access.

- Breakpoint capability if trace is full.

- 1K x 32-bit real-time trace (address, data, ctrl).

- 6 different modes to configure trace access by combining 3 event conditions.

- Selective trace recording capability.

- 2-Trigger output capability.

- Can use as an external clock source either the on-probe oscillator, or an external source via the front panel input.

- 4 probe inputs to display application signals in the trace.

## 4.2     Specific Features of the ST7MDT6-EMU2B

The features described below are specific to the **ST7MDT6-Active Probe**:

- Clock source selection.

- The application power supply follower allows this emulator to run with application $V_{CC}$ from 3.5 V to 5.5 V. When the probe is not connected to an application board or if the application board isn't powered (application $V_{CC} <$ 3.5 V), the $V_{CC}$ default value is 3.5 V.

- All I/O pins of the ST72C171 are directly interconnected to the user application board. No buffers or protective devices are inserted on the probe.

- Probe mechanical dimensions: 123 mm x 100 mm (5"x 4").

- SDIP32 or SO34 adapter height: 20 mm (0.8").

## 4.3      Emulator Architecture

The ST7MDT6-EMU2B emulator is composed of 2 parts:

- The **ST7 HDS2** (Hardware Development System) contains all of the common resources necessary to emulate any ST7 device (such as memory and the link interfaces with the PC). This board is connected to the PC via a parallel link and to the second part by two 50-pin connectors.

- The **ST7MDT6-Active Probe** contains the specific resources necessary to emulate the ST72C171 MCU, and is used as a link between the ST7 HDS2 and your application.
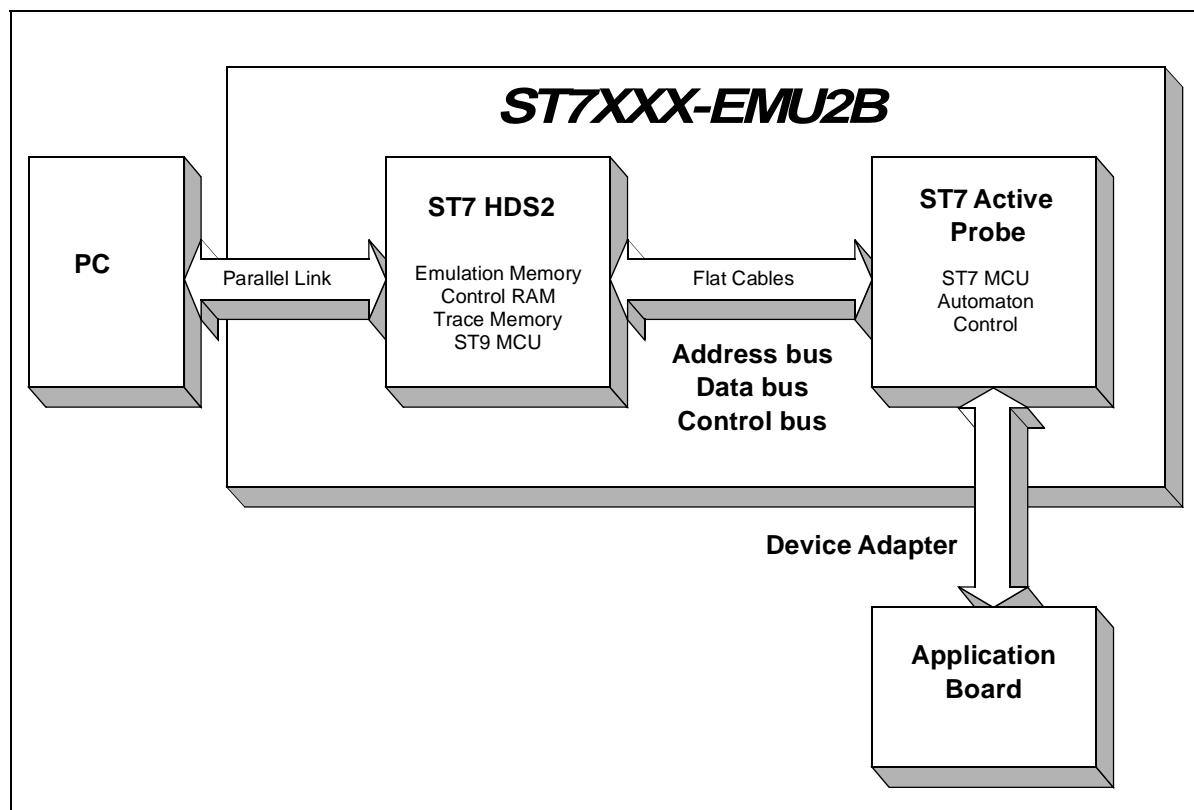


### 4.3.1     ST7 HDS2 Hardware

The hardware functions provided by this component are listed below:

- **HDS2 CPU:** Used to control of the ST7XXX-HDS board and manage common HDS2 features such as the parallel link with the PC.

- **RAM memory**:

    - 64K bytes for ROM and RAM emulation.

    - 64K bytes as break points control and Mapping.

- 1K x 32-bit as trace memory.

• **Hardware breakpoint control logic** to manage breakpoints from the 16-bit address bus.

• **Logical analyser control logic** to manage sophisticated recording and break events in the trace.

• **PC link:** parallel interface for communication with PC.

• **ST7MDT6-Active Probe interface**—3 buses connect the ST7 HDS2 to the **ST7MDT6-Active Probe**:

- Address bus (16-bit) of the ST7 emulation chip used for RAM addressing and trace.

- DATA bus (8-bit) of the ST7 emulation chip.

- Control bus to manage ST7MDT6-Active Probe hardware-like breakpoint features.
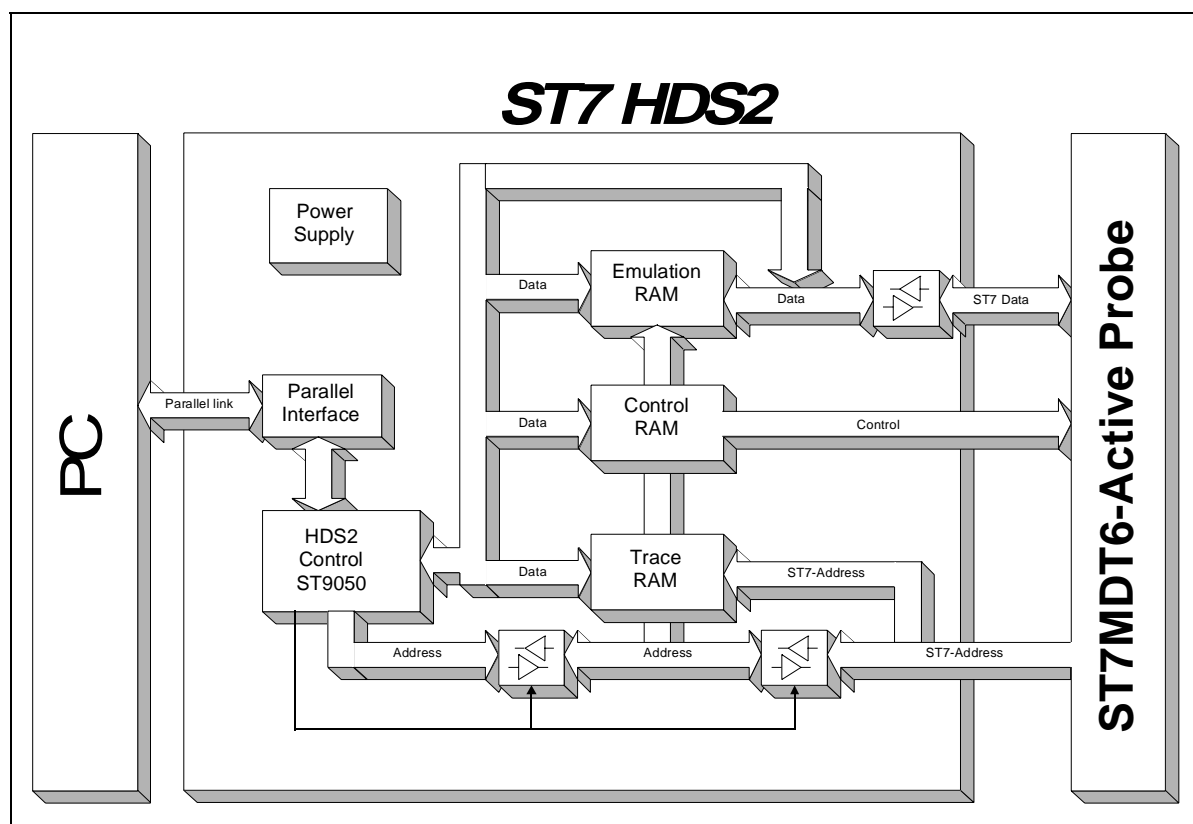


### 4.3.2      ST7MDT6-Active Probe Hardware

The hardware functions provided by this component are:

- **Probe Emulation MCU:** This is an ST7 microcontroller similar to those of the emulated target device(s), which runs in emulation mode. It acts as the ST7 core and gives access to all on-chip peripherals.

- **Control logic:** Control logic is provided to manage the software execution by the user (i.e. program running and halting).

- **Application $V_{CC}$ follower:** The probe emulation MCU is supplied with the same voltage as the application (i.e. must be in the range 3.5 V to 5.5 V).

- **ST7 HDS2 interface:** All of the communication buses connecting the active probe to the ST7 HDS2 board are buffered:

  - ST7 Address bus (16-bit) of the ST72C171 in emulation mode.

  - Data bus (8-bit) of the ST72C171 in emulation mode.

  - ST7 emulation chip control bus for trace recording, breakpoints and memory mapping.
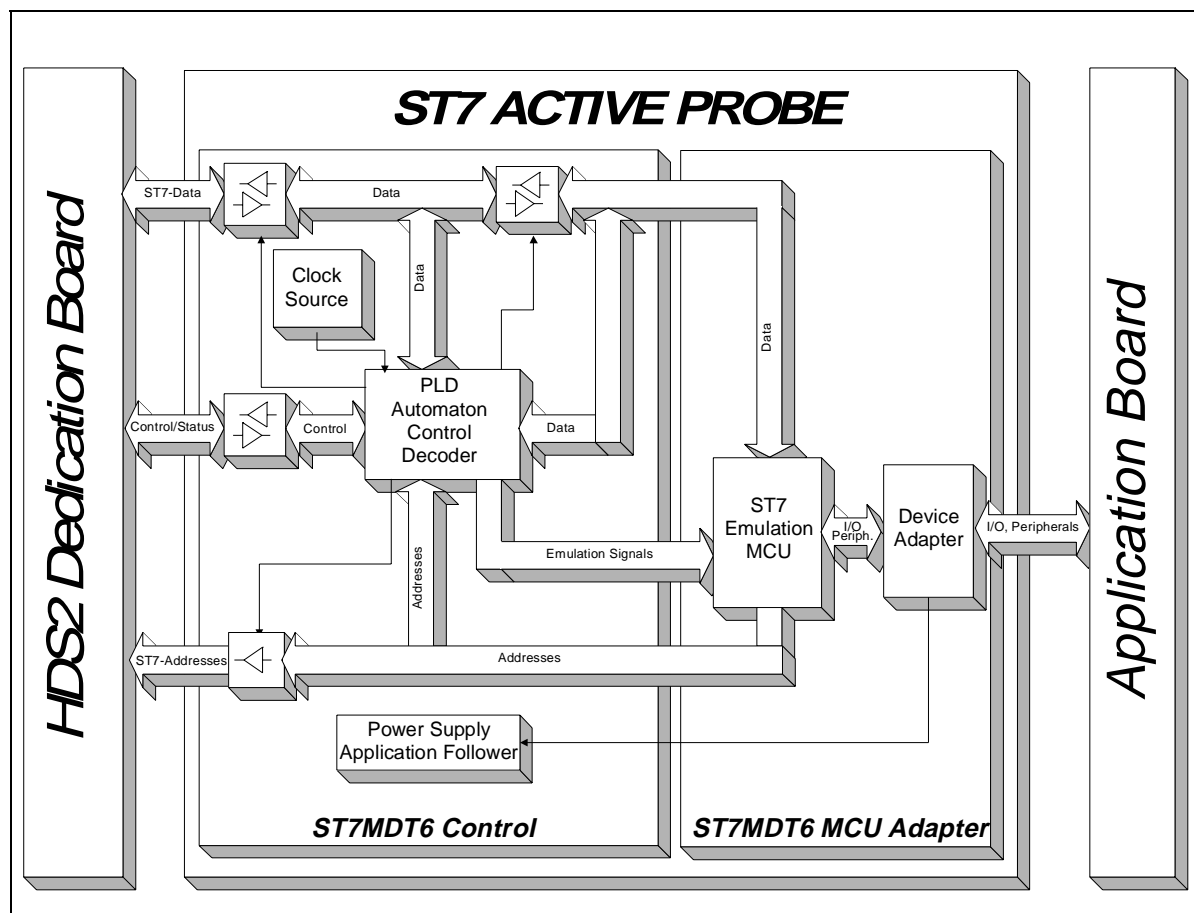
## 4.4       Output Triggers

Your ST7 HDS2 emulator has two output triggers, OUT1 and OUT2. The OUT1 and OUT2 outlets are available via SUB-click connectors located on the front panel of the ST7 HDS2 emulator box.
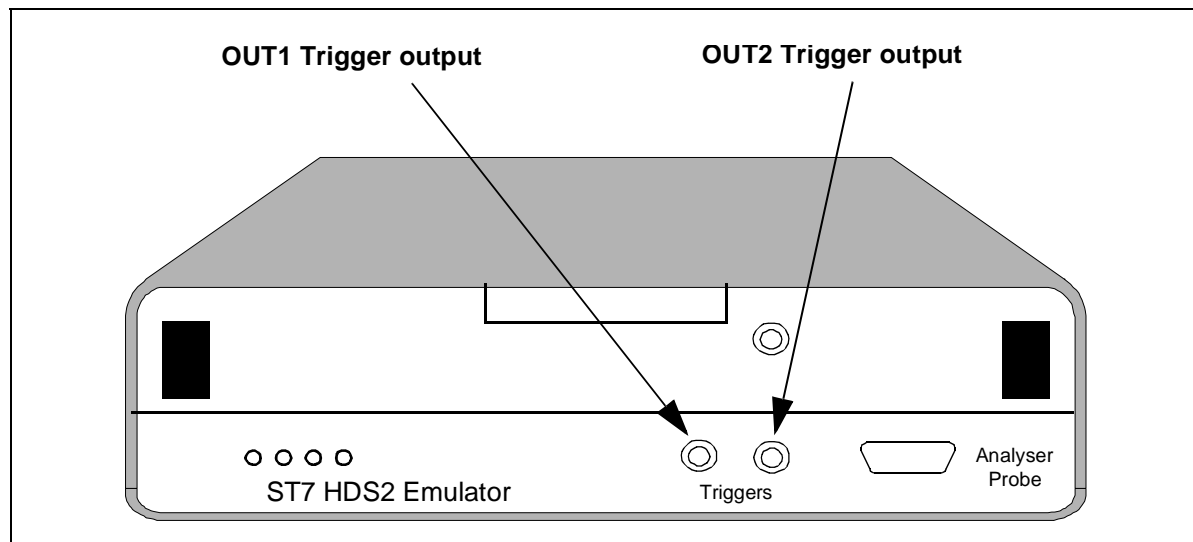


You can program the output signals to these triggers using ST7 Visual Debug:

**1**  From the main menu in ST7 Visual Debug, select **View>Hardware Events**. The Hardware Events window will open in your workspace.
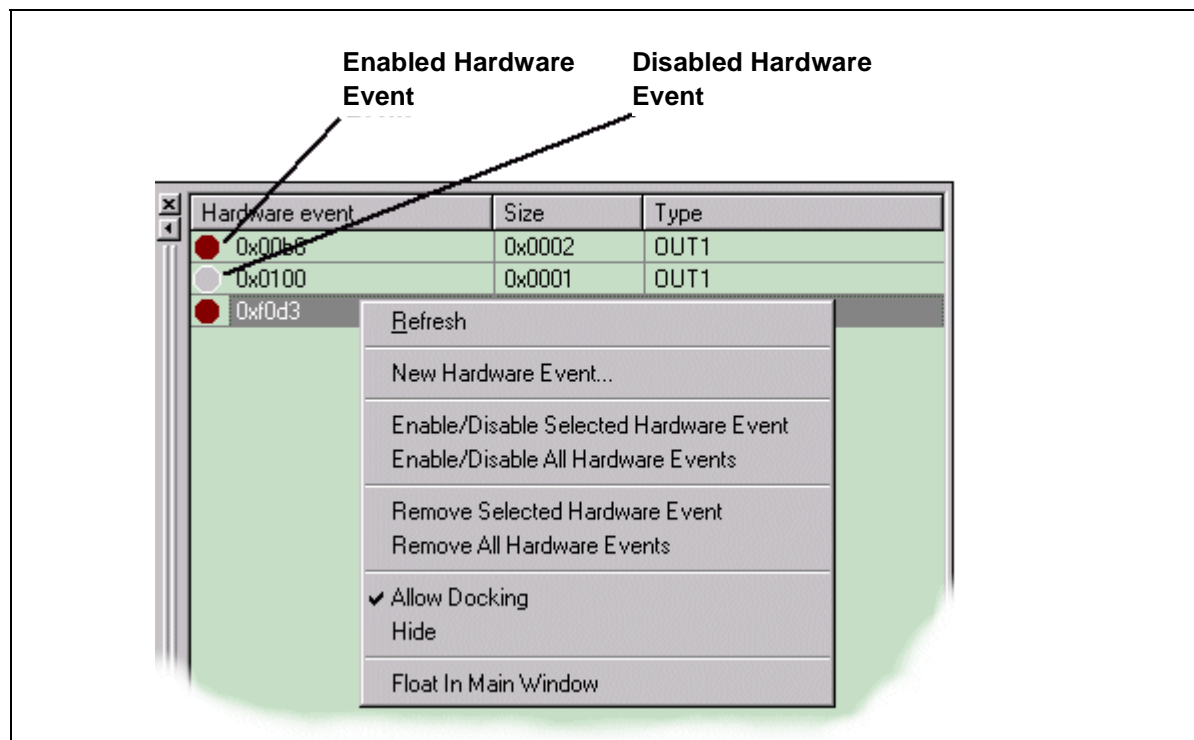


**2**  Right-click the mouse while the mouse pointer is anywhere in the **Hardware Events** window.

**3**  Choose **New Hardware Event** from the contextual menu. The **Hardware event settings** dialog box will open as below.
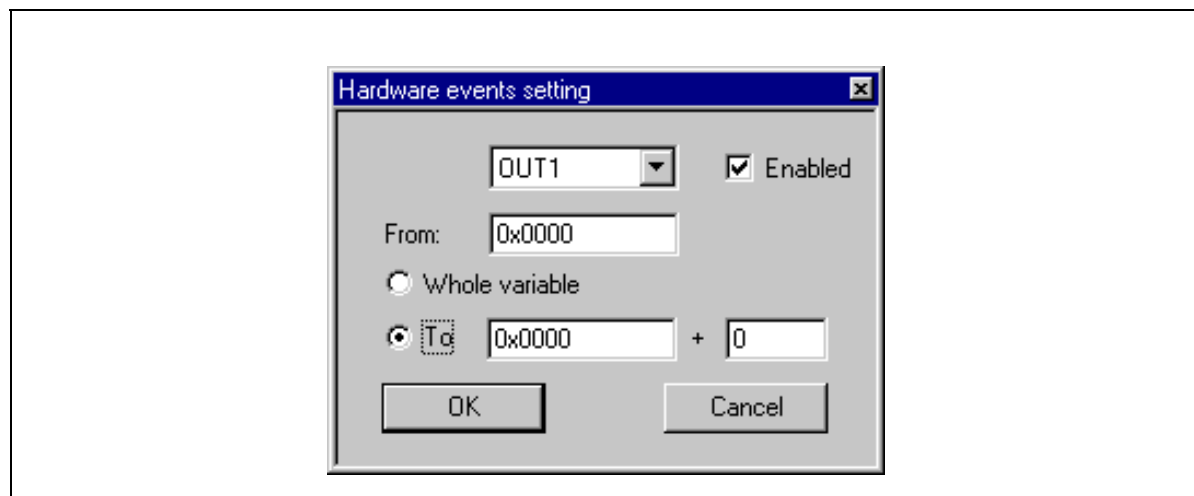
**4**   Choose the trigger output that you wish the signal to be sent to (i.e. OUT1 or OUT2) and check the **Enabled** box.

**5**   You may trigger output signals by setting an event on any of the following:

-   **a whole variable**—creating an *event for synchronization*, which enables you to preset the pulse synchronization for external equipment connected to the output trigger.

-   **a single address**—also creates an *event for synchronization* (see above).

-   **a range of addresses**—creating an *event to measure time*, which enables you to measure the time elapsed during a subroutine execution.

A positive impulse is emitted on OUT1 and OUT2 when a specific condition is met. This impulse lasts for one Clock cycle.

## 4.5   Analyser Probe Input Signals

The ST7 HDS2 allows you to use 4 external input signals (TTL level). These signals are on pins 6,7,8,9 of the Analyser Probe connector located on the front panel of the emulator as shown below.
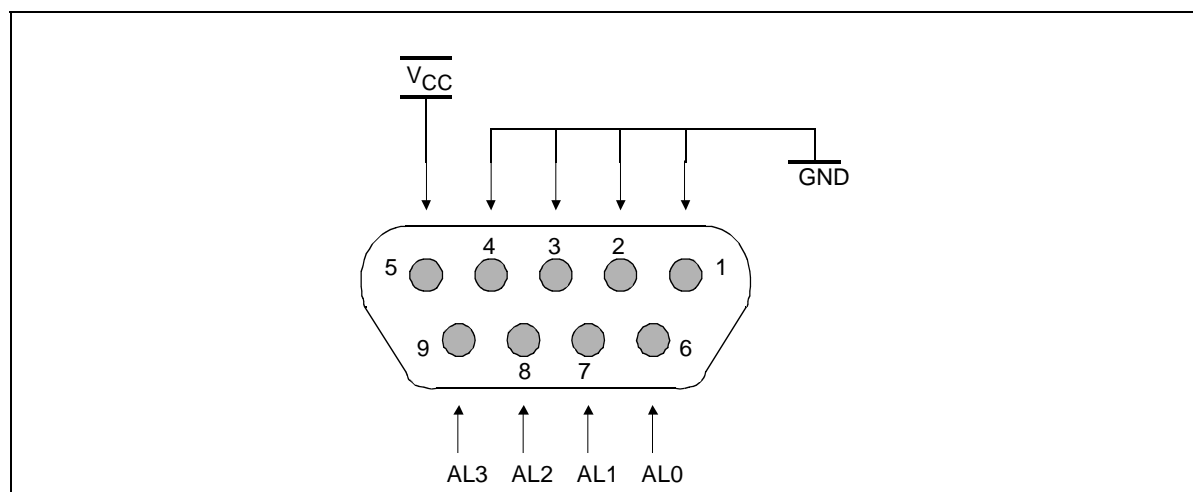


You can view these analyser probe inputs using ST7 Visual Debug. From the main menu, select **View>Trace**. The input signal values are listed under the Sig column (AL3..0).

ST7 Visual Debug's **Logical Analyser** allows you to use these input signals to define trace filtering or output trigger events. From the main menu, select **Tools>Logical Analyser** to open the dialog box. A full description of how to use this facility to control trace recording or trigger output signals is given in the ST7 Visual Debug online help.

A rainbow-colored cable will also be delivered to connect your application to these inlets. Each red connector is to be connected to your signal. Each black connector is to be connected to the reference ground for the signal.

Colors are attributed as follows:

- AL0 is to be taken between the RED (signal) and BROWN (ground) wires.

- AL1 is to be taken between the YELLOW (signal) and ORANGE (ground) wires.

- AL2 is to be taken between the BLUE (signal) and GREEN (ground) wires.

- AL3 is to be taken between the GREY (signal) and PURPLE (ground) wires.

## 4.6    Front Panel LEDs

Four LEDs on the front panel of the HDS2 box indicate the state of the development tool during emulation:

- **Power (Green)**—indicates that the 5 V power supply is ON.

- **Run (Yellow)**—indicates that the ST7 is running (not in RESET, WFI and HALT mode).

- **System (Red)**—not used with this emulator.

- **Reset (Red)**—not used with this emulator.
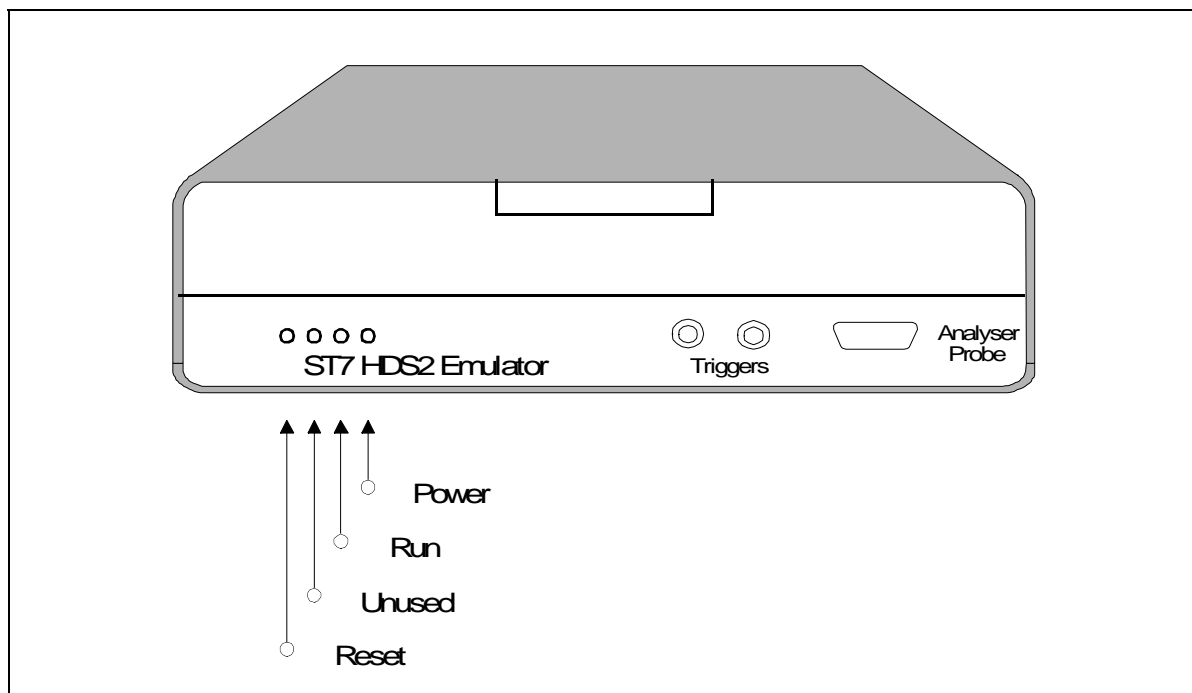
**4.7        On-Chip Peripherals**

You can configure certain on-chip peripherals in ST7 Visual Debug's **MCU Configuration** dialog box (refer to *Configuring the MCU* on page 34) so that the emulator accurately emulates your target device.

The on-chip peripheral options supported by the emulator are:

**4.7.1      CLOCK**

The emulator can work with three clock sources (Max. Frequency =  16 Mhz, Min. Frequency = 5 Mhz):

• **16 MHz** internal oscillator.

• **8 MHz** internal oscillator.

An **external clock** source. The external clock source signal is provided by the user via the SUB-click connector located on the ST7MDT6-Active Probe as shown below.



ST7MDT6-Active Probe
(ref.:DB397)

SUB-Click External Clock Connector
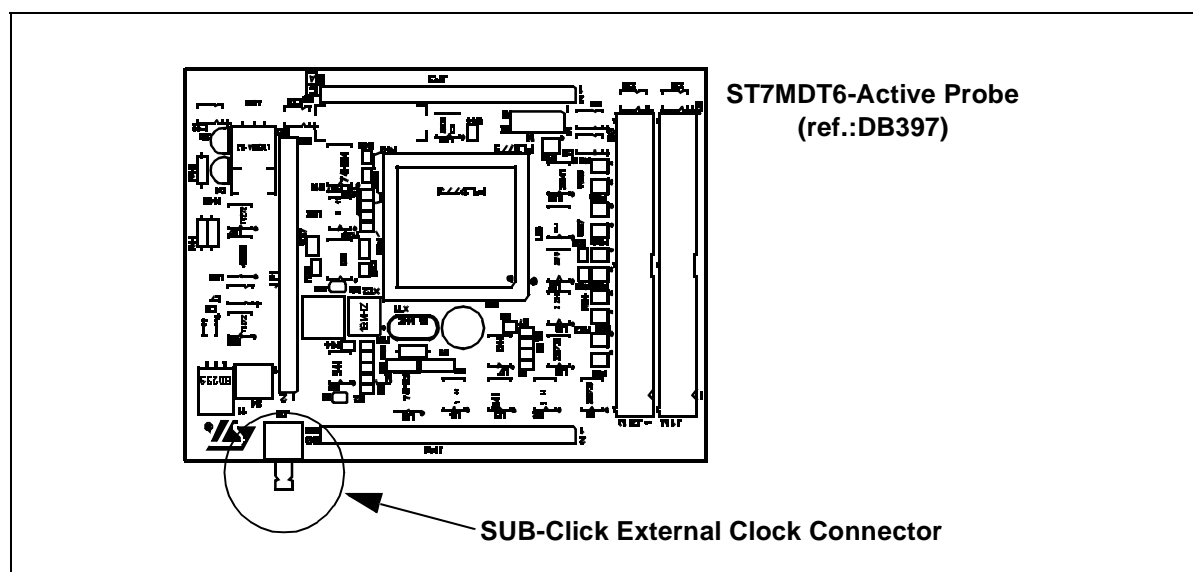
The SUB-click **external** clock connector on the probe can be used with the two SMB to BNC connectors provided in the emulator package. However, if additional connectors are needed, you can purchase them at the dealers or manufacturers listed in the section entitled *Hardware spare parts* on page 60. (This list is not exhaustive.)

*Note:*       *Refer to Emulation Functional Limitations and Discrepancies on page 50 for information on the application clock available on your emulator.*

### 4.7.2    CLOCK_FILTER

When **ON**, the internal cell clock filter of the emulation MCU is enabled. Refer to the datasheet for your MCU for more information on the internal clock filter.

### 4.7.3    LVD_STATUS

Low voltage detection is not currently available with this emulator, so this option is always **OFF**.

### 4.7.4    WGD+HALT

Allows you to determine if a reset is generated when entering HALT mode while the watchdog is active. There are two options: **Reset**, where a reset is generated when entering Halt mode if the watchdog is active, and **No Reset**, where no reset is generated when entering Halt mode.

### 4.7.5    WATCHDOG

This option allows you to choose whether the watchdog timer is enabled by software or by hardware.

When the **Software** option is chosen, the watchdog has to be enabled by software. When the **Hardware** option is chosen, the watchdog is always enabled.

Refer to the datasheet for your ST7 MCU for more information on the watchdog timer.

## 4.8    Emulation Functional Limitations and Discrepancies

### 4.8.1    Power Supply

The application supply follower allows this emulator to run with an application $V_{DD}$ ranging between **3.5 V to 5.5 V**. If the application isn't powered, or the $V_{DD} < 3.5$ V, the power supply is maintained at 3.5 V. If your application is powered by a voltage greater than 5.5 V, the emulator will limit this value internally to 5.5 V.

### 4.8.2    Probe Clock

There is a footprint for a built-in 16 MHz oscillator on the ST7MDT6-Active Probe, to allow you to emulate your application clock by installing the same clock oscillator on the probe. However, this feature is not supported by the current version of the emulator.

### 4.8.3    LVD_STATUS

This function is not supported on the ST7MDT6-EMU2B emulator. (The LVD is always disabled.)

# APPENDIX A: TROUBLESHOOTING

## A.1        Identifying the Problem

| IF YOU RECEIVE THE FOLLOWING: | THEN... |
|---|---|
| **Error Message** (when starting the STVD7 for HDS Emulator): **"No message received from emulator."** | Ensure that:<br>• The parallel cable is connected between the development board and one of the PC's parallel ports (LPT1 or LPT2). Note that the use of switch boxes between the parallel port connector of your PC and the development board are not recommended.<br>• The development board is powered on.<br>• The parallel cable used is the one supplied with the kit by STMicroelectronics.<br><br>If none of the above items has been overlooked, this may mean that your parallel port connection needs to be reconfigured.<br><br>Please refer to the section entitled *Changing the Parallel Port Setup on Your PC*, below. |
| **Error Messages** (when starting the STVD7 for HDS Emulator): **"Communication error with EMULATOR board."** <br>or<br>**"SYSTEM ERROR DETECTED by EMULATOR BOARD: RESET CPU."** | Ensure that:<br>• The flat cables linking the ST7MDT6-Active Probe and the emulator box are properly connected.<br>• The selected configuration file matches the connected ST7MDT6-Active Probe configuration.<br>If it doesn't, from within ST7 Visual Debug, open the **MCU Configuration** dialog box by selecting ***Tools>MCU Configuration*** from the main menu.<br>Choose the correct MCU target device in the dropdown list, then click **OK** to save your changes. |

## A.2        Changing the Parallel Port Setup on Your PC

Under certain circumstances, you may receive the following error message:

> **"Connection Error (LPT1/LPT2): Interconnection failure. Verify your input/ output cable."**

This may mean that the setup of the LPT1 or LPT2 port on your PC is not compatible with the ST7MDT6-EMU2B.

To set up the port correctly:

**1**   Shut down and restart your PC in order to enter the BIOS setup.

**2** Follow the messages displayed on the screen and when prompted, press the key required to enter the BIOS setup (usually a function key or the ESC key).

**3** Select the parallel ports menu. (This may be listed under I/O ports.)

**4** Change the Mode of the LPT port that you have connected the development board to (i.e. either LPT1 or LPT2) to one of the following compatible modes, according to the following table:

| Operating System | Compatible Parallel Port Modes |
|---|---|
| Windows 95 | ECP, EPP, Bidirectional or Centronics |
| Windows 98 | EPP, Bidirectional or Centronics |
| Windows NT4 | ECP, EPP, Bidirectional or Centronics |

**5** Save your changes and exit the BIOS setup.

## A.3 Running the Hardware Test

The **Hardware Test** in ST7 Visual Debug lets you check that your emulator is correctly connected, configured and working. You can test components of the development board individually, or all at the same time.

If problems occur during debugging (such as bad debugger responses and unexpected behavior), you should check for hardware problems using the Hardware Test function, and if any are detected, contact your STMicroelectronics sales representative (*Product Support* on page 59).

You may open the **Hardware Test** dialog box by:

- selecting, from the Main Menu, **Emulator>Hardware Test**

- clicking on the Hardware Test icon  in the **Emulator** toolbar.

**Warning:** **Be cautious in performing a Hardware Test on the emulator while an application is open. The opened application WILL BE corrupted by the hardware testing process. If you find that your application has been corrupted, simply close the application, and reopen it.**

The **Hardware Test** dialog box shows a list of different tests that can be performed on the emulator.

Check the box of each test that you wish to perform (they are all checked by default) and click **Apply** to start the hardware test.
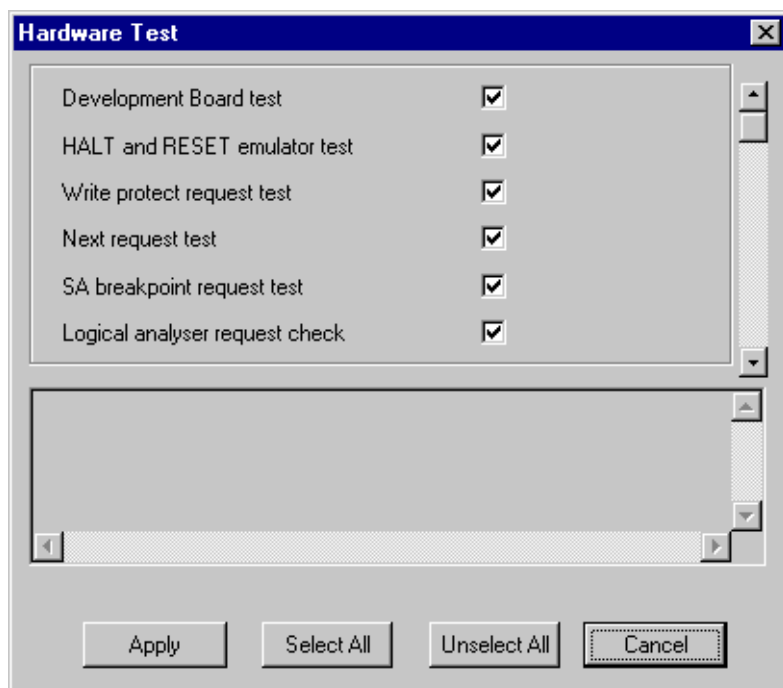


The Hardware tests will be performed one by one, and the results summarized in the dialog box as shown on the right:
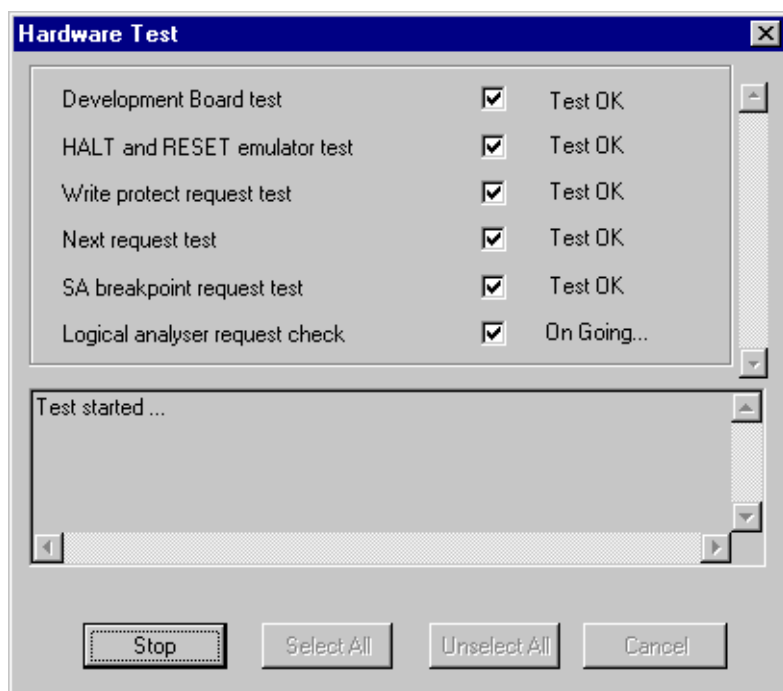
# APPENDIX B: GLOSSARY

**Application Board**

This is the printed circuit board onto which you wish to connect the target ST7 MCU. It should include a socket or footprint so that you can connect the application board to your emulator or development kit using the probe and the appropriate device adapter. This allows you to emulate the behavior of the ST7 MCU in a real application in order to debug your application program.

**Device Adapter**

Device adapters are included in your emulator kit to allow you to connect the emulator to your application board. The type of device adapter depends on the target device's packaging. Many MCUs come in more than one different package, and you should therefore use the device adapter that corresponds to the type of package you have chosen for your application.

**DIL**

Dual In Line. Designates a type of device package with two rows of pins for thru-hole mounting. Sometimes also called DIP (Dual In-line Package).

**ECP**

Extended capabilities port communication standard.

**EPP**

Enhanced parallel port communication standard.

**Footprint**

Designates the dimensions of the location of a component on a printed circuit board or in a socket. It depends on the number of pins, their size, type and positioning. The footprint of each ST7 device is specified in the datasheet in the section titled *Package Mechanical Data.*

**MCU**

Microcontroller Unit. Otherwise referred to as the "target device" throughout this manual. This is the core product (or family of products) for which the Development Kit is designed to act as an emulator and programming tool. In general terms, an MCU is a complete computer system, including a CPU, memory, a clock oscillator and I/O on a single integrated circuit.

**ST7MDT6-Active Probe**

A printed card having connector pins that allow you to connect the Emulator to the MCU socket of the user application board. Using the active probe allows the HDS2 emulator to function as if it were the target device embedded in your application. The probe is connected to the emulator by two flat cables.

**PC (Program Counter)**

The program counter is the CPU register that holds the address of the next instruction or operand that the CPU will use.

**RC network**

Resistor-capacitor network.

**SDIP**

Serial Dual In-line Package.

**SO**

Small outline. Designates a type of device package with two rows of pins for SMD or socket mounting. For example, SO34 designates a 34-pin device of this package type.

**ST7 Visual Debug (STVD7)**

A graphic debugger software package that allows you to debug applications destined for the ST7 family of MCUs, either using a built-in simulator function, a Development Kit or an HDS2 Emulator.

**Target Device**

This is the ST7 MCU that you wish to use in your application, and which your emulator or development kit will emulate for you.

**User Application Board**

Designates your application board.

# PRODUCT SUPPORT

If you experience any problems with this product or if you need spare parts or repair, contact the distributor or ST sales office where you purchased the product:

## Getting prepared before you call

Collect the following information about the product before contacting ST or your distributor:

1  Name of the company where you purchased the emulator kit.

2  Date of purchase.

3  Order Code: Refer to the side of your emulator kit box. The order code will depend on the region for which it was ordered (i.e. the UK, Continental Europe or the USA).

4  Serial Number: The serial number is located on the rear panel of the emulator box.

5  Target Device: The sales type of the ST7 microcontroller you are using in your development.

## Contact List

*Note:*      *For* **American and Canadian customers** *seeking technical support the US/Canada is split in 3 territories. According to your area, contact the following sales office and ask to be transferred to an 8-bit microcontroller Field Applications Engineer (FAE).*

### Canada and East Coast

STMicroelectronics
Lexington Corporate Center
10 Maguire Road, Building 1, 3rd floor
Lexington, MA 02421
Phone: 781-402-2650

### Mid West

STMicroelectronics
1300 East Woodfield Road, Suite 410
Schaumburg, IL 60173
Phone: 847-517-1890

**West coast**

STMicroelectronics, Inc.
30101 Agoura Court
Suite 118
Agoura Hills, CA 91301
Phone: 818-865-6850

**Europe**

France (33-1) 47407575
Germany (49-89) 460060
U.K. (44-1628) 890800

**Asia/Pacific Region**

Japan (81-3) 3280-4120
Hong-Kong (852) 2861 5700
Sydney (61-2) 9580 3811
Taipei (886-2) 2378-8088

**Software updates**

You can get software updates from the ST Internet web site http://mcu.st.com. For information on firmware and hardware revisions, call your distributor or ST using the contact list given above.

**Hardware spare parts**

Most of the hardware you will require is included in the emulator  kit. However, some special applications may require additional parts, such as connecting an external clock.

Below is a list of manufacturers and dealers of SMB and BNC connectors that can be used with our product.

**European manufacturers:**

*Manufacturer*: Radiall
*Dealer*: Radialex
*Phone*: (+33) - 4 - 72 - 35 - 31 - 72.

The EXTERNAL clock male connector on the emulation probe has the following commercial reference:

-    In SMB range: Number 114665.

Adaptable Female connectors to this connector are:

- SMB upright range
  Number114005 for cable 2,6.
  Number114003 for cable 4,2.
  Number114009 for cable 3,8.

- SMB kneed range
  Number114165 for cable 2,6.
  Number114163 for cable 4,2.

- SMB to BNC range
  Number191214. Adapter SMB female / BNC male.
  Number191215. Adapter SMB female/ BNC female.

**USA manufacturers:**

*Manufacturer*: R-Tek
411 Quentin Road
Palatine, IL 60067
Phone: (847) 934-7900
Fax: (847) 934-7946

Adaptable female connectors parts numbers:

- CCAX00168-2: cable length 2 ft, with SMB plug to BNC plug.

- CCAX00168-3: cable length 3 ft, with SMB plug to BNC plug.

# Index

# Index